In Proceedings of The 2th International Conference on Semantics, Knowledge, and Grids(SKG2006),
Guilin, China, Oct. 31 - Nov. 3, 2006

Abox Inference for Large Scale OWL-Lite Data*

Xiaofeng Wang, Jianbo Ou, Xiaofeng Meng, Yan Chen
Renmin University of China
{zzuwxf,oujianbo,xfmeng,chenyan8}@ruc.edu.cn

Abstract

Abox inference is an important part in OWL data management. When involving large scale of instance data, it can not be supported by existing inference engines. In this paper, we propose efficient Abox inference algorithms for large scale OWL-Lite data. The algorithms can be divided into two categories: initial inference and incremental inference. Initial inference is used in situation where only raw data exists in storage system, and for this category we propose Rule Static Association Based (RSAB), Rule Dynamic Association Based (RDAB) and Rule Grouped-Sorted Based (RGSB) inference methods. Incremental inference algorithm is used in situation where large volume inference data exists in storage system, and for this category we extend the initial inference algorithm and propose Rule Pattern-Sharing Based(RPSB) method. At last, extensive experiments show that our methods are efficient in practice.

1. Introduction

OWL[1] (Web Ontology Language) is proposed by W3C and it is used to describe Web resource. OWL-Lite is a sublanguage of OWL. It was disigned for easy implementation and to provide users with a functional subset that will get them started in the OWL. Inference for OWL-Lite can be divided into two categories: Tbox and Abox. Tbox inference concerns Ontology and generally can be supported by existing inference engines[6, 7, 12, 5]. Abox inference focuses on large scale instance data, and can not be supported by existing inference engines.

As the research of semantic web is becoming more and more popular,many semantic repositories have been developed,which can be divided into three categories based on the storage strategy they use: RDB (Relational Database)-based[8, 10, 9],File system-based[4] and Memory-based[3].For the RDB has been studied extensively these years, RDB-based systems are in the majority, like Sesame[8], DLDB-OWL[10], RStar[9] and so on. Sesame provides a general storage interface and implements storage method on MySQL, Oracle and so on. Sesame aims at storage for RDF files, and doesn't take the feature of OWL data into consideration.From experiment, we observe that large number of rules are needed to express complete OWL semantics and loading process is also very inefficient for doing complete inference based on such rules. Therefore, it can't be used to manage large scale OWL data. DLDB-OWL uses MS Access as its persistent platform and uses inference engine FaCT[7]. It declares high performance for large scale OWL data, but has limited inference ability. From experiment we observe that DLDB-OWL cannot get any answers for some queries. OWLim[3] is a typical memory-based system. It supports more semantic rules than any other systems. OWLim uses Sesame's general storage interface and has higher performance than Sesame's own memory-based storage module.But when do query and inference processing, all data will be read from hard disk into memory. Because OWLim supports most of the semantic rules in OWL Lite, we use it as benchmark of query completeness in our experiment. To support large scale semantic data management, we develop HStar system, which is built on file system and do query and inference processing on physical storage model. In this paper, we mainly concern the inference engine in HStar system.

This paper discusses Abox inference for large scale OWL-Lite data and divides the algorithms into two categories: initial inference and incremental inference. Initial inference is used in situation where only raw data exists in storage system. Because inference about one single rule has high computation cost, it is important to remove redundant rule inference. From this point, we propose Rule Static Association Based (RSAB), Rule Dynamic Association Based (RDAB) and Rule Grouped-Sorted Based

*Supported by the National Natural Science Foundation of China under Grant Nos.60073014, 60273018; China National Basic Research and Development Program's Semantic Grid Project No. 2003CB317000; the Key Project of Chinese Ministry of Education under Grant No.03044 ;Program for New Century Excellent Talents in University

(RGSB) inference algorithms.Compared to basic inference algorithm, RSAB algorithm removes some redundant rule inference procedures. But there still exist some redundant procedures. RDAB takes data set and rules together into consideration, any redundant procedures can be removed by this algorithm. The whole Rules of OWL-Lite Abox inference has lattice-like relation,based on which we can group rules and sort such groups. Do inferences based on Group-Sorted Rules can make inference more focused and then reduce the number of data and pattern matching.

Incremental inference algorithm is used in situation where large volume inference data existing in storage system.So we propose Rule Pattern Sharing Based(RPSB)increment inference algorithm to reduce number of data and pattern matching. Compared to old data, new data is generally small in scale. So buffering new data in the sharing patterns can avoid redundant inference and reduce number of data and pattern matching.

The contributions of this paper are:

(1) We summarize general Abox inference methods for large scale OWL-Lite data and divide the algorithms into two categories: initial inference and incremental inference.

(2)For initial dataset, we propose Rule Static Association Based (RSAB), Rule Dynamic Association Based (RDAB) and Rule Grouped-Sorted Based (RGSB) inference algorithms besides basic inference algorithm.

(3) We extend initial inference algorithm and design Pattern-Sharing Based incremental inference(PSB) algorithm to handle incremental inference.

(4) We present an extensive performance evaluation of different inference algorithms and analyze the results in detail.

2. Preliminaries

Before introducing any specific inference algorithms, first we will explain some symbols used in this paper. [2]describes the features of OWL-Lite, according to the semantic of which we list rules related to Abox inference shown in table 1.

The basic unit of rule is binary relation related to Ontology data and ternary relation related to Instance which are described by RDF syntax. We call them rule pattern, written by P, whose formal format is: P = P_{onto} |P_{Ins}. P_{Ins} = (S P O), in which S, P and O can be variables. P_{onto}=Rel(Class |Property |Class, Property), in which Rel(Class) describes the relationship between Class, Rel(Property) describes relationship between Property. Rel(Class, Property) describes the relationship between Class and Property. Inference Rule in Abox can be formalized as below:

R = [P_{onto},] P_{Ins-1},..., P_{Ins-n} :- P_{Ins-x}, the part before symbol ':-' is called rule premise, and the part after symbol ':-' is called rule conclusion. If we use { $V_1, V_2, \ldots, V_m$ }to represent variable sets of P_{Ins-1},...,P_{Ins-n}, then the variables which belong to P_{Ins-x} is in the subset of { $V_1, V_2, \ldots, V_m$ }.

1	EquivalentClass(u,v),(x type u):-(x type v)
2	SubPropertyOf(u, v),(x u y):-(x v y)
3	FunctionalProperty(p),(u p y),(u p x):-(y sameAs x)
4	InverseFunctionalProperty(p),(x p u),(y p u):-(x sameAs y)
5	TransitiveProperty(p),(x p y),(y p z):-(x p z)
6	SymmetricProperty(p), (x p y) :- (y p x)
7	InverseOf(p, q), (x p y) :- (y q x)
8	EquivalentProperty(p, q), (x p y) :- (x q y)
9	Domain(p, c), (x p y) :- (x type c)
10	Range(p, c), (x p y) :- (y type c)
11	SubClassOf(u, v), (x type u) :- (x type v)
12	(u sameAs v) :- (v sameAs u)
13	(u sameAs v), (u p x) :- (v p x)
14	(u sameAs v), (x p u) :- (x p v)
15	AllValuesFrom(c, p, d), (x p y), (x type c) :- (y type d)

Table 1 inference rule in OWL-Lite

3. Initial Abox Inference Algorithms

To begin, we give the definition of complete inference dataset, which defines complete results that do not omit any inference procedures, then we talk about basic rule inference, at last we will explain rule association based inference algorithms.

3.1. Basic Inference Method

Before introducing any inference methods,first,we want to explain inference complete dataset,for it is critical to the final results.

Definition 1: Complete Inference Dataset *Given the dataset D which is declared by users explicitly, we call D* the complete inference dataset after inference procedure, if and only if it satisfies the following condition: no new data can be generated in D* using any rule.*

According to the above definition, we can get D* from D using the rules shown in table 1 repeatedly. We first discuss inference methods focusing on single rule.

The rule premise consists of several patterns, and the association between patterns is connected by the same variable. For example, the association between patterns (a p b) and (b q c) is connected by variable b. Generally speaking, there are two methods to compute all instance data from the rule premise. One is to use join method and the other is navigation. The former method is to match the pattern with the data, then for each pattern we can get a result set, at last these sets are joined to get the final set. The latter method is to match a certain pattern with the data, then use its result set to query the pattern associated with it, this

procedure continues until all the patterns have been handled. In most cases,the navigation method is better than the join method.When many variables in the pattern are unknown,the join method would probably match all instances in the dataset,which is unaccepted for large scale data.

Based on definition 1, we propose single rule inference algorithm using navigation method below.({I} represents value sets of instance data matching the pattern P_i)

Algorithm 1: SingleRuleReason(R,D)

```
begin
    select rule R whose pattern P_i has known element;
    {I}← Query(P_i);
    repeat
        Select a pattern P_j which has least unknown
        elements associated to P_i;
        Substitute corresponding variables in P_j with
        the value in instance sets I and obtain P_j';
        {J} ← Query(P_j');
        if J= NULL then
            delete corresponding data from {I};
        else
            combine I and J and get {I,J};
    until the rule premise of R has been scanned once
    ;
    Substitute corresponding variable in rule
    conclusion of R with the value in instance set I and
    obtain result set D';
    foreach item e in D' do
        if e doesn't exists in D then
            insert it into D; return TRUE;
    return FALSE;
end
```

Based on Algorithm 1,it is easy to think of a method which do inference straightforward.When new data is generated by using certain rule, in order to assure the completeness of the inference, we need do inference for all rules over again, for newly generated data may match the rule premise, and the rule may lead to new data. When we have no prior association in advance, it is an effective way to do inference iteratively to ensure the completeness of the result.

Algorithm 2: BasicReason(D)

```
begin
    BOOLEAN hasNewData ← FALSE;
    repeat
        foreach R_i in RuleSet do
            hasNewData
            ← SingleRuleReason(R_i, D);
    until hasNewData is FALSE ;
end
```

3.2. Association Based Inference Methods

Algorithm 2 doesn't consider the association of rules, so we may do redundant inference. When a certain rule R generates new data, which triggers other rules {R'}, then R and {R'} have association relationship. If we can get all R and its association rules{R'}, we only need to do inference on {R'} to avoid redundant inference.

Definition 2: Pattern Match *P_{Ins-A}(a p b) match P_{Ins-B}(x q y) if and only if x is a variable or a=x, and q is a variable or p=q, and y is a variable or b=y.*

Above definition ignores the case when pattern (x p u) matches pattern(x type u), for this kind of match is related to instance data, we call it *conditional pattern match*. If the rule conclusion of R_1 conditional matches the rule premise of R_2, we call R_1 conditional association match R_2.

Based on the above discussion, we can improve basic rule inference algorithm.The main idea is to provide a waiting rule set.In an iterative inference procedure, if rule R generates new data,then we add rules that are associated to R to waiting rule set,after that, we check whether new generated data satisfies conditional association rules, if so, we add the rules to the waiting rule set too. In next iterative inference procedure, we do inference according to rules in the waiting rule set. This method can avoid redundant inference. Improved algorithm called Rule Static Association Based Inference (RSAB)algorithm is shown as Algorithm 3.

Algorithm 3: ReasonAlgorithm_1(D)

```
begin
    Put all rules into WaitSet;
    clear WaitSet_1;
    repeat
        foreach R_i in WaitSet do
            if SingRuleReason(R_iD) then
                Add rules associated to R_i to
                WaitSet_1
                if new generated data satisfies
                conditional association rules then
                    add rules to WaitSet_1;
        WaitSet = WaitSet_1;
        clear WaitSet_1;
    until WaitSet is NULL ;
end
```

Algorithm 3 avoids some redundant inference procedures, but not thorough.For example, in table 2, new data generated by rule 1 can trigger rule 2, but when matching instance data, the case is not always true. Say, rule 1 generates (x' type v') , when it is applied to rule 2, we can get SubProperty(type v), (x' type v'), if SubProperty(type, v) has no corresponding instance data, rule 2 won't generate new inference procedure, so we need not add rule 2 to WaitSet. Determination of dynamic association between rules

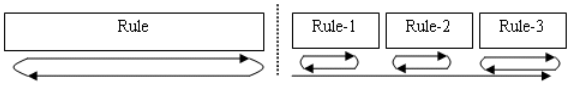

Figure 1 RGSB and RAB inference algorithm

is similar to that of conditional association rule. We use instance data to test if there exists matched data for non-matched pattern.Now we give Rule Dynamic Association Based Inference algorithm shown as Algorithm 4.

Algorithm 4: ReasonAlgorithm_2(D)

```
begin
    put all rules into WaitSet;
    clear WaitSet_1;
    repeat
        foreach R_i in WaitSet do
            if SingleRuleReason(Ri,D) then
                Add rules associated to R_i which
                match instance data generated by R_i
                Based on table 1;
                if new generated data satisfies
                conditional association rules then
                    add rules to WaitSet_1;
        WaitSet = WaitSet_1;
        clear WaitSet_1;
    until WaitSet is NULL ;
end
```

Algorithm 3 and Algorithm 4 describe association based inference algorithms, which avoid redundant inference procedure by association relationship, but do not consider inference order.

Let's come back to table 1 again, and assume that there are only rule 5, 6 and 9, if we do inference in such order [{5,6} → {9}] , we can assure that rule 9 need only be executed once, for newly generated data by rule 9 won't trigger rule 5 and rule 6, so we can put them in one group, and do inference before rule 9, or else we should do inference for rule 9 repeatedly. for example, if we do inference in such order [{5}→{9}→{6}], and if rule 6 generates new data, it will trigger rule 9 , probably trigger rule 5, we need a new run of inference [{5}→{9}→{6}].When we group rules in advance, the inference iteration faces rule group, not the whole rule sets.Figure 1 illustrates the difference between Rule Grouped-Sorted Based inference algorithm and Rule Association Based inference algorithm.

After the discussion, we propose rule grouped-sorted based inference algorithm in Algorithm 5:

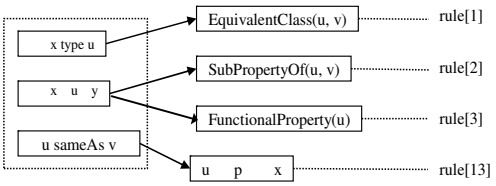

Figure 2 Pattern Sharing Structure

Algorithm 5: ReasonAlgorithm_3(D)

```
begin
    assume the order after grouping is
    [{R1}→{R2}→...→{Rn}];
    for i ← 1 to n do
        do inference on rule set{R_i};
        using ReasonAlgorithm_2(D);
end
```

4. Incremental Abox Inference Algorithms

In section 3, we described Initial inference, now let's consider another scenario: inference complete data has already existed in storage system, now new data comes and needs to be inserted into the system. If we use the methods introduced in section 3 directly, we will do redundant inference on old data repeatedly, for those algorithms do not differentiate new data and old data.

When new data and old data coexist,obviously,the old data is inference complete and we want to avoid inference procedure on it. But inference on new data may have relation with the old data, for new data and old data probably match certain rule premise.

In figure 2,we show design pattern sharing structure. we test each pattern on the new data, if rules share the same pattern, the times that rules match on new data can be greatly reduced. Take rule {1,2,3,13}in table 1 for example, in the rule premise P_{Ins}, the common pattern is[(x u y)x, u, y are variables].

Box drawn with dashed line in figure 2 contain all the rule premise(P_{Ins}), we can see that rule 2 and rule 3 share the same pattern (x u y). When rules are represented in such structure, we need only test the patters enclosed by the box in dashed line, then we can complete the inference procedure along the pointer of the pattern.we propose Rule Pattern Sharing Based(RPSB) incremental inference algorithm shown as Algorithm 6.

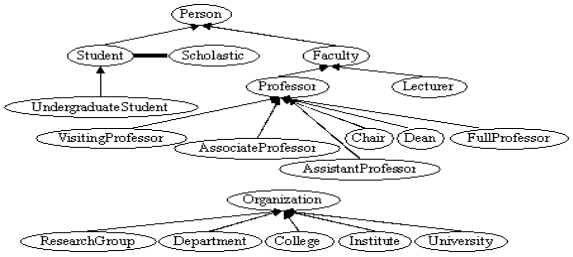

Figure 3 Class Hierarchy

Algorithm 6: ReasonAlgorithm_4(Δ D, D)

begin
  Assume the sharing patterns are $\{P_1, \ldots, P_m\}$;
  clear WaitSet;//waiting to be inferred
  foreach d_i *in* Δ *D* **do**
    foreach P_x *in* $\{P_1, \ldots, P_m\}$ **do**
      if d_i *matches* P_x **then**
        Assign corresponding variables in P_x and the pattern P_x pointing to;
        for each pattern P_x pointing to, query in new and old data and get inference result;
        if *new data exists in inference result* **then**
          add association rules based on ReasonAlgorithm_2(D) to the WaitSet;
end

5. Experiments

We now experimentally evaluate the techniques presented in this paper. First, we present the performance of different algorithms in the initial dataset. Second,In the case of incremental inference, we compare the performance between RPSB and RGSB.

Dataset: We use Benchmark[11] developed by LEHIGH university, which provides test data and its corresponding Ontology, and also defines 14 typical queries reflecting OWL semantic features. The details of the dataset will be discussed later.

Figure 3 and figure 4 separately depict the Class and Property hierarchy of the Benchmark.

Environment: All experiments were conducted on a Pentium IV 1.7GHz machine with 512M memory, and 40 G hard disk , running the WindowXP. We conducted our experiments on HStar system, which is an extention of OrientX developed by Renmin university of China. All experiments were repeated 10 times and the average processing time was calculated.

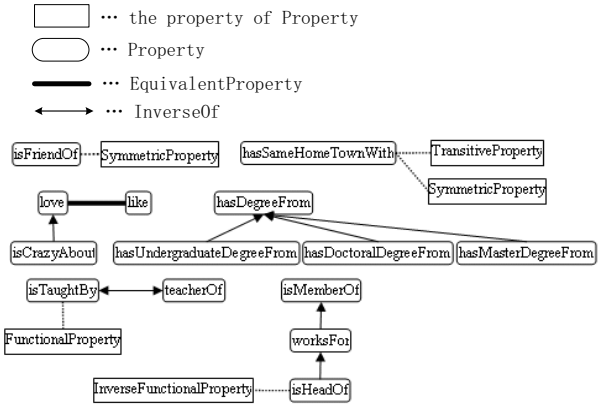

Figure 4 Property Hierarchy

Queries: We design the following 6 queries in order to test the completeness of inference. These queries cover all the inference rules.
Query-1. (? rdf:type Professor)
Query-2. (? rdf:type Student)
Query-3. (uri rdf:type ?)
Query-4. (uri isFriendOf ?)
Query-5. (uri hasSameHomeTownWith ?)
Query-6. (uri love ?)

In this experiment, we use three instance datasets and their corresponding Ontology. The number of triples and file size are shown in table 2:

dataset	tripple numbers	file size(KB)
Ontology	428	42
Instance 1	10694	992
Instance 2	19321	2483
Instance3	30181	4658

Table 2 Triple numbers and file size

Because the basic inference algorithm is based on integrity definition, its results can be used as the standard sets. table 3-5 shows the results of different inference algorithms on the above 6 queries, from which we can make a conclusion that the initial inference algorithms are complete and correct.(BI represents Basic Inference, None represents No Inference)

Query	None	BI	RSAC	RDAB	RGSB
Query-1	0	36	36	36	36
Query-2	0	666	666	666	666
Query-3	2	8	8	8	8
Query-4	2	5	5	5	5
Query-5	2	4	4	4	4
Query-6	1	1	1	1	1

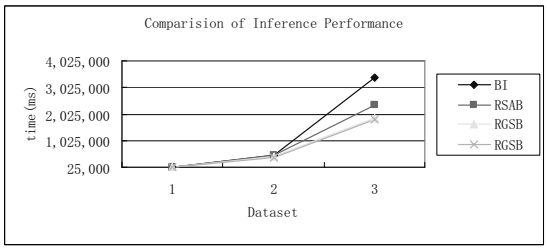

Figure 5 efficiency comparisons among different inference algorithms

Table 3 completeness comparison on dataset 1

Query	None	BI	RSAC	RDAB	RGSB
Query-1	0	44	44	44	44
Query-2	0	666	666	666	666
Query-3	2	18	18	18	18
Query-4	3	14	14	14	14
Query-5	5	70	70	70	70
Query-6	0	1	1	1	1

Table 4 completeness comparison on dataset 2

Query	None	BI	RSAC	RDAB	RGSB
Query-1	0	52	52	52	52
Query-2	0	666	666	666	666
Query-3	5	18	18	18	18
Query-4	9	730	730	730	730
Query-5	7	141	141	141	141
Query-6	0	18	18	18	18

Table 5 completeness comparison on dataset 3

Next we compare the efficiency of the four inference algorithms, figure 5 shows the performance of the different approaches. RGSB performs best, while the performance of basic inference declines sharply as the dataset increases.

At last, we evaluate the performance two incremental inference algorithms, which are RGSB and RPSB.We evaluate the performance from two aspects: processing time and results completeness.We omit the completeness evaluation due to limited space.

Figure 6 is about the processing time between RGSB and RPSB. Not surprisingly, RPSB inference algorithm outperforms RGSB.

6. Conclusion

We have presented algorithms for Abox inference on large OWL-Lite data in initial dataset, including basic inference, RSAC, RSAB and RGSB. From the experiment, we can see that these inference algorithms are complete, among which RGSB performs best. When large amount of inference data exists, we provide incremental inference,

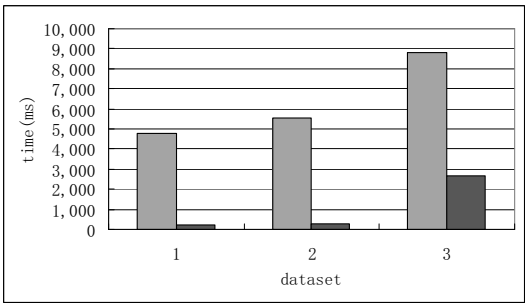

Figure 6 processing time between RGSB and RPSB

which not only ensure the inference completeness but also improve the efficiency. At last,extensive experiments show that our methods are efficient in practice.

References

[1] Owl,web ontology language. *http://www.w3.org/2004/OWL*.

[2] Rdf,resource description framework. *http://www.w3.org/RDF/*, 2004.

[3] D. O. D. M. Atanas Kiryakov. Owlim - a pragmatic semantic repository for owl. *In Proc. of Int. Workshop on Scalable Semantic Web Knowledge Base Systems*, 2005.

[4] P. G. David Wood and T. Adams. Kowari: A platform for semantic web storage and analysis. *In WWW*, 2005.

[5] V. Haarslev and R. Moller. High performance reasoning with very large knowledge bases: A practical case study. *In B. Nebel, editor, Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 161–166, 2001.

[6] V. Haarslev and R. R. Moller. A core inference engine for the semantic web. *In Workshop on Evaluation on Ontology-based Tools*, pages 27–36, 2003.

[7] I. Horrocks. The fact system. *In Automated Reasoning with Analytic Tableaux and Related Methods International Conference*, pages 27–30, 1998.

[8] A. K. J. Broekstra and F. Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. *In Proc. of the 1st International Semantic Web Conference(ISWC)*, pages 54–68, 2002.

[9] Y. P. L. Z. Li Ma, Zhong Su and T. Liu. Rstar: An rdf storage and query system for enterprise resource management. *In CIKM*, 2004.

[10] Z. Pan and Heflin. J. dldb: Extending relational databases to support semantic web queries. *In Workshop on Practical and Scalable Semantic Systems*, 2003.

[11] Z. P. Y. Guo and J. Hefin. An evaluation of knowledge base systems for large owl datasets.

[12] T. F. Youyong Zou and H. Chen. F-owl: an inference engine for the semantic web. *In Book Formal Approaches to Agent-Based Systems*, pages 238–248, 2004.