Incremental Maintenance Of Dynamic Datalog Programs - Extended Abstract -

Raphael Volz[1,2], Steffen Staab[1] and Boris Motik[2]
(1) Institute AIFB, University of Karlsruhe
{volz,staab}@aifb.uni-karlsruhe.de
(2) WIM, Forschungszentrum Informatik (FZI)
{volz,motik}@fzi.de

1 Introduction

This paper is an extended abstract of [5], which discusses the incremental maintenance of materialized ontologies in a rule-enabled Semantic Web. The reader may wonder how this relates to Datalog, a language that has been proposed in the deductive database context. However, the semantics underlying Web ontology languages can often be realized by translating the ontology into appropriate Datalog programs. Therefore, our solution for incrementally maintaining materialized ontologies actually builds on the incremental maintenance of dynamic Datalog programs.

Materialization generally allows to speed up query processing and inferencing by explicating the implicit entailments which are sanctioned by the rules of the program. The complexity of reasoning with Datalog is thereby shifted from query time to update time. We assume that materialization techniques will frequently be important for the Semantic Web to achieve a scalable solutions, since read access to is predominant in a Web setting. Central to materialization are maintenance techniques that allow to incrementally update a materialization when changes occur.

We present a novel solution that allows to cope with changes in rules and facts for Datalog programs. To achieve this we extend a known approach for the incremental maintenance of views (intentional predicates) in deductive databases. We show how our technique can be employed for a broad range of existing Web ontology languages, such as RDF/S and subsets of OWL.

Our technique can be applied to a wide range of ontology languages, namely those that can be axiomatized by a set of rules in Datalog with stratified negation. The challenge that has not been tackled before is dealing with updates and new definitions of rules. However, our solution extends a declarative algorithm for the incremental maintenance of views [4] that was developed in the deductive database context.

2 Datalog and Web ontology languages

Since the early days of the Semantic Web, many systems have tried to reason with Web ontology languages using rule-based systems, e.g. Triple [3]. To do so, a particular Web

ontology language is axiomatized via a static set of rules which capture the semantics specified for a particular ontology language. For example, the following rule is part of the Datalog axiomatization of RDF/S:

t(A,subClassOf,C) :- t(B,subClassOf,C), t(A,subClassOf,B). *(rdfs8)*

This rule assumes that ontology and knowledge base is stored in a single ternary predicate t, i.e. the extension of t stores all triples that constitute a particular RDF graph. In order to apply our techniques we have to distinguish assertions from entailments. We achieve this by turning the predicate t into a completely intensional predicate (view) that is derived from the explicitly asserted information. Hence, asserted RDF triples constitute a separate extensional predicate, say t_{Ext}, and t is derived from t_{Ext} via a rule:$t(X, Y, Z)$:- $t_{Ext}(X, Y, Z)$.

The set of rules is typically not immutable. With the advent of higher layers of the Semantic Web stack, i.e. the rule layer, users can create their own rules. Hence, we are facing a scenario where not only base facts can change but also the set of rules. This requires the ability to maintain dynamic Datalog programs. Besides support for a rule layer, this ability is also required for approaches where the semantics of the ontology language is not captured via a static set of rules but instead compiled into a set of rules. Such an approach is for example required by Description Logic Programs (DLP) [1], where OWL ontologies are translated to logic programs. Other implementations of knowledge representation languages, e.g. O-Telos and F-Logic , have also been achieved via such a compilation. Consequently, a change to the ontologies class and property structure will result in a change of the compiled rules. Again, it is necessary to be able to maintain a materialization in case of such a change.

3 Changing facts

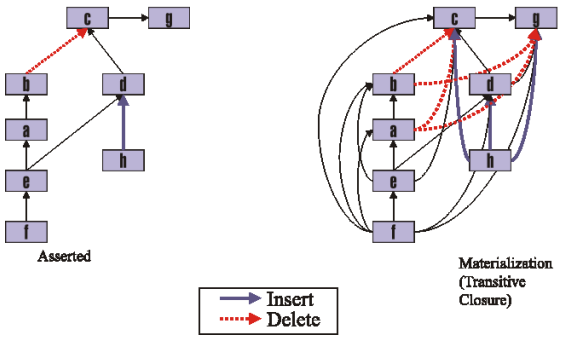

Figure 1: Changes to a RDF/S taxonomy and its materialization

Let's consider the effects of adding and removing a subClassOf relationship in the taxonomy presented in Figure 1 with respect to the presented rule, which implements the transitive closure of the `rdfs:subClassOf` relationship, and a rule that links the intensional predicate t with ground facts. We can easily see that : (a) b is no longer a subclass of c, viz. the $t_{ext}(b, \text{subClassOf}, c)$ fact is deleted. (b) h is asserted to be a subclass of d, hence a $t_{ext}(h, \text{subClassOf}, d)$ fact is inserted.

However, this has the following consequences to t: It eliminates the links between (a,c), (a,g), (b,c) and (b,g). Since the facts $t_{ext}(d, \text{subClassOf}, c)$ and $t_{ext}(e, \text{subClassOf}, d)$ exist, alternative derivations also exist for the links (e,c), (e,g), (f,c) and (f,g). These alternative derivations have to be taken into account in our approach. The insertion yields three new derivations namely links between (h,c), (h,d) and (h,g).

Several algorithms (e.g. [4, 2]) have been presented for the incremental maintenance of views (or intensional predicates) in the deductive database context. The most common procedure is to compute the changes (differentials) to views in three steps: Firstly, to overestimate the consequences of deletions, so a super set of the facts that are eventually deleted is computed for deletion. Secondly, a rederivation step prunes those computed deletions from the set of deletions for which alternative derivations (via some other rules defining the view) exist. Thirdly, the consequences of insertions to extensional predicates are added to the view, if applicable.

Our approach is based on the approach of [4], which realizes the DRed (delete and rederive) algorithm presented in [2] in a purely declarative way. Basically, the original program is rewritten into a maintenance program which is evaluated instead of the old program, viz. a set of maintenance rules is generated.

4 Changing rules

Our core contribution is to update a materialization if the definition of intensional predicates changes, viz. rules that define the predicate are added or removed. Our solution has two main components. Firstly, the materialization itself has to be maintained. Secondly, the maintenance rules for a predicate p themselves have to be maintained. We will only discuss the first component in the following.

Adding and removing rules requires the reevaluation of all other rules that define the same predicates. It does not suffice to simply change the maintenance rules which are generated for dealing with changes in facts, since these maintenance rules (in their original form) are only activated when changes to the extension of a predicate occur. Therefore we have to generate additional maintenance rules. Our solution is based on the creation of a temporary predicate p^{Temp}, which is used to calculate the extension of a predicate p using the changed set of rules. Hence, p^{Temp} is axiomatized using the updated rule set that defines a predicate p. Of course, this has to incorporate some changes into the rule set, for example self-references of the predicate have to be substituted by the temporary predicate.Then, p^{Temp} is incorporated into the maintenance program us-

ing a new set of maintenance rules and interacts with the existing machinery that deals with changes in facts.

Our approach has that serious drawback that a change in rules might require to recompute the whole program, for example if all rules define a single triple predicate only as it is often done to axiomatize RDF/S. Naturally, this situation corresponds to the simple strategy of recomputing a materialization from scratch.

For this case we propose the *selection-based optimization*, which improves the maintenance process by limiting the part of the database which takes part in the evaluation. Our optimization splits predicates into multiple predicates depending on split points. Split points are given by constants that occur at a certain argument position of a predicate. A Datalog program is then transformed into an equivalent program, such that all references to p where a split point occurs are replaced by the new split predicates. After splitting, we can ignore all rules that do not define those split predicates which are affected by the change and use our old machinery without having to recompute everything.

5 Conclusion

We have presented the ideas underlying our approach to incremental maintenance of dynamic Datalog programs. We invite the reader to have a closer look to our full paper [5], which presents the technical details and the results of a first performance evaluation, which demonstrates the feasibility of our approach. Our open-source Java implementation is part of the KAON Datalog engine, which is available for download at http://kaon.semanticweb.org/.

We believe that materialization will become a central technique for achieving scalability in Distributed Semantic Systems such as presented by the Semantic Web. We have shown how our approach can be used with current means for specifying semantics in the Semantic Web. As we present a generic solution, future developments, e.g. for the rule layer of the Semantic Web, are likely to benefit from our technique as well.

References

1. B. Grossof, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: Combining Logic Programs with Description Logic. In *Proceedings of WWW 2003*, Budapest, Hungary, May 2003.
2. A. Gupta, I.S. Mumick, and V.S. Subrahmanian. Maintaining views incrementally. In *ACM SIGMOD Conference on Management of Data*, 1993.
3. Michael Sintek and Stefan Decker. TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In *International Semantic Web Conference (ISWC)*, June 2002.
4. Martin Staudt and Matthias Jarke. Incremental maintenance of externally materialized views. In *VLDB'96*, pages 75–86, 1996.
5. Raphael Volz, Steffen Staab, and Boris Motik. Incremental Maintenance of Materialized Ontologies. In *2nd Int.Conf. on Ontologies and Databases (ODBASE)*, Sicily, Italy, October 2003.