

Description Logic Reasoning for Dynamic ABoxes

Christian Halaschek-Wiener, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur
Maryland Information and Network Dynamics Laboratory

8400 Baltimore Ave., College Park, MD, 20740 USA

{halasche, evren, aditya}@cs.umd.edu, bparsia@isr.umd.edu

Abstract

Various data sources on the Web tend to be highly dynamic. This is evident in services that provide streaming content, and in Web portals where the user community publishes content to the site. Today's Description Logic (DL) reasoners have only been considered for relatively static knowledge bases. Given that there has been recent interest in using DL languages for representing Web content, this work aims to provide more efficient DL reasoning techniques for frequently changing ABoxes. More specifically, we investigate the process of incrementally updating the tableau completion graphs used for reasoning in the expressive Description Logics $\mathcal{SHOQ}(\mathcal{D})$ and $\mathcal{SHIQ}(\mathcal{D})$, which correspond to a large subsets of the W3C standard Web Ontology Language, OWL-DL. We present an algorithm for updating completion graphs under both the addition and removal of ABox assertions. We also provide an empirical analysis of the update procedure through an implementation in the Pellet OWL-DL reasoner.

1 Introduction

Data sources on the Web continue to evolve and are moving to a more fluctuating environment with streaming video, audio, RSS feeds, etc. Many community based websites additionally allow their users to post comments, articles, etc., providing an environment in which the content of the portal is constantly being updated.

Recently, there has been interest in providing formal representation of Web content, which can then be processed using automated reasoning techniques. Due to data sources that produce fluctuating data, there exists a variety of description logic reasoning use cases which require frequent updates at the assertional level. These include prominent web services frameworks (e.g., OWL-S) that use description logics for service discovery and matchmaking, where devices register or deregister their

descriptions (and supporting ontologies) quite rapidly. Additionally, Semantic Web portals often allow content authors to modify or extend the ontologies which organize their site structure or page content. Currently, RSS feeds are currently encoded in RDF; as this and similar services move to more expressive representations, reasoners will need to be able to process continuously changing content. Lastly, one of the common uses of description logic reasoners is in ontology editors. Most editors do not do continuous reasoning while one is editing, relying on an analogue of the *edit-compile-test* loop of most programming environments.

While there exists such use cases for reasoning under update, today’s description logic reasoning services have been developed considering relatively static knowledge bases. In this paper, we investigate the process of incrementally updating tableau completion graphs created during consistency checking in the expressive description logics $\mathcal{SHIQ}(\mathcal{D})$ and $\mathcal{SHOQ}(\mathcal{D})$, which correspond to a large subset of the W3C standard Web Ontology Language, OWL-DL. We present an algorithm for updating completion graphs under both the addition and removal of ABox assertions; this provides a critical step towards reasoning over fluctuating/streaming data, as it has been shown that in KBs with substantially sized ABoxes, consistency checking can dominate reasoning time. We also provide an empirical analysis of the optimizations through an experimental implementation in an open source OWL-DL reasoner, Pellet.

2 Background

In this section, we describe we briefly discuss tableau algorithms for description logic reasoning. Due to space limitation, the syntax and semantics of the Description Logic \mathcal{SHOIN} is omitted, however a detailed discussion can be found in [6].

DL tableau-based algorithms decide the consistency of an ABox A w.r.t a TBox T^1 by trying to construct (an abstraction of) a common model for A and T , called a *completion graph* [6]. Formally, a completion graph for an ABox A with respect to T is a directed graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \neq)$. Each node $x \in \mathcal{V}$ is labeled with a set of concepts $\mathcal{L}(x)$ and each edge $e = \langle x, y \rangle$ with a set $\mathcal{L}(e)$ of role names. The binary predicate \neq is used for recording inequalities between nodes. This graph is constructed by repeatedly applying a set of *expansion rules*.

For ABox updates, we consider addition and deletion of individual equality and inequality assertions $x = y$ and $x \neq y$, concept assertions $C(x)$ and role assertions $R(x, y)$. Updating ABox assertions in the presence of a TBox and an RBox (especially for an expressive language such as the ones considered in this paper) brings up several issues with the semantics. Such problems have been studied extensively in the literature for updating (propositional and first-order) knowledge bases [11, 8].

The main problem of defining an update semantics is the *ramifications* of axioms in the KB. Consider a TBox $T = \{C \sqsubseteq D\}$ and an ABox $A = \{C(a), D(a)\}$. Now

¹by TBox, we are additionally referring to all axioms for properties

if we want to update A by removing the assertion $D(a)$ then one might consider also removing the assertion $C(a)$ because it entails (along with the TBox) the assertion that we are trying to remove.

For the purposes of this paper we use a straight-forward update semantics, which we call *Edit Semantics*. Intuitively, *Edit Semantics* can be described as an update in which all new ABox assertions are directly added (or removed) to the KB and no further changes/revisions are made. As the above example shows removing an assertion from the ABox under these semantics will not guarantee that the removed assertion will not be entailed anymore. Furthermore, an addition of a new axiom may cause inconsistencies in the KB as no additional changes can be made besides the addition or removal invoked by the update. This semantics might be a little unintuitive (especially from a belief revision point of view) but real world use of the update semantics is directly present in ontology editors and many document-oriented servers, e.g. Semantic Web Service repositories. For example, an OWL-S Web Service description can be seen as a set of ABox assertions and publishing/retracting this service description to/from a repository would be typically done under *Edit Semantics*. Edit semantics additionally aligns with the

Formula based update semantics defined in [11]. Formally, we describe this semantics as:

Definition 1 (*Edit Semantics*) *Let S be the set of assertions in an initial ABox A . Then under Edit Semantics, updating S with an ABox addition (resp. deletion) α , written as $A + \alpha$ (resp. $A - \alpha$), results in an updated set of ABox axioms S' such that $S' = S \cup \{\alpha\}$ (resp. $S' = S \setminus \{\alpha\}$).*

3 Update Algorithm

The goal of the algorithm presented here is to update a previously constructed completion graph in such a way that it is equivalent to a completion graph that *could* be rebuilt from scratch after the update has been performed in the ABox. The algorithm presented here is applicable for the Description Logics *SHOQ* [4] and *SHIQ* [5], as the difference between the two completion algorithms is independent of the update algorithm. As stated earlier, in situations where the ABox is large, this can provide a substantial performance improvement.

First, we abstractly define the update algorithm as $UPDATE(G, \alpha)$ (shown in Figure 1), which takes as input a completion graph, G and an update, α and returns a new completion graph. For purpose of this work, let us additionally define a *completion rules application* function $CR(K)$, which takes a knowledge base K as input and returns the set of all possible completion graphs obtained by nondeterministically applying all expansions rules to K . We also define a *rule firing sequence* function $RFS(G)$, that returns a set of all possible tableau expansion rule firings that can result in the generation of the completion graph G . Lastly when discussing the nodes

and edges in a completion graph, we may collectively refer to them as the *components* on the graph.

When updating ABoxes, we separate the task into handling two cases, *additions* and *deletions*, which are explained in detail below.

3.1 ABox Additions

Conceptually, tableau algorithms for \mathcal{SHOQ} [4] and \mathcal{SHIQ} [5] can be thought of as incremental in nature - *expansion rules* are applied in a non-deterministic manner to labels in the completion graph. Hence, new ABox assertions can be added even after previous completion rules have been fired for existing nodes and edges in the graph. After the addition, expansion rules can subsequently be fired for the newly added nodes, edges and their labels.

In order to update a completion graph upon the addition of a type assertion, $C(x)$, the algorithm first checks if the individual exists in the completion graph. If $x \notin \mathcal{V}$, then x is added to \mathcal{V} . Then C is added to $\mathcal{L}(x)$, if it does not already exist. If an individual inequality relation, $x \neq y$, is added to the completion graph, the algorithm checks if $x \in \mathcal{V}$ and $y \in \mathcal{V}$. If either do not exist, then they are added to the graph. Additionally, if the $x \neq y \notin \mathcal{E}$, then it is added. Alternatively, if an individual equality relation, $x = y$, is added to the completion graph, the algorithm checks if $x \in \mathcal{V}$ and $y \in \mathcal{V}$. If either do not exist, then they are added to the graph. Additionally, x and y are merged. Lastly, if a role, $R(x, y)$, is added to the ABox and $\langle x, y \rangle \notin \mathcal{E}$, then $\langle x, y \rangle$ is added to \mathcal{E} and R is added to $\mathcal{L}(\langle x, y \rangle)$. If however, $\langle x, y \rangle \in \mathcal{E}$ but $R \notin \mathcal{L}(e)$, then R is added to $\mathcal{L}(x)$.

After the graph has been updated, the completion rules are reapplied to the updated completion graph, as the update may induce additional expansion rules to be fired. We note here that if there has previously been a deletion update, then previously explored branches which had a clash must be re-explored as the deletion could have removed the axiom which caused the clash.

3.2 ABox Deletions

When updating a completion graph under ABox axiom removals, *components* (nodes, edges, labels, etc.) in the graph that correspond to the removed axiom cannot be simply removed. This is because as completion rules are applied, newly added portions of the graph are dependent on the presence of the original axioms in the KB. By deleting an ABox assertion, components of the graph that are dependent on that assertion need to be updated as well.

In order to account for this, we propose using axiom pinpointing [7], which tracks the dependencies of graph components on original source axioms from the ontology through the tableau expansion process. More specifically, the application of the expansion rules triggers a set of *events* that change the state of the completion graph, or

the flow of the algorithm. In [7] a set of change events is defined as follows:

- **Add** $(C, \mathcal{L}(x))$ represents the action of adding a concept C to the label of a node x , i.e. the operation $\mathcal{L}(x) \leftarrow \mathcal{L}(x) \cup \{C\}$.
- **Add** $(R, \mathcal{L}(\langle x, y \rangle))$ represents the addition of a role R to the label of an edge $\langle x, y \rangle$, i.e., $\mathcal{L}(\langle x, y \rangle) \leftarrow \mathcal{L}(\langle x, y \rangle) \cup \{R\}$.
- $x = y$ is the action of *merging* the nodes x, y . A detailed description of the *merge* operation can be found in [6].
- $x \neq y$ stands for the addition of an inequality relation between two nodes x, y , where x, y , i.e. $\neq \leftarrow \neq \cup \{\langle x, y \rangle\}$.

In order to record the changes on the completion graph, [7] introduces a *tracing function*, which keeps track of the asserted axioms responsible for changes to occur. The *tracing function*, τ , maps each event $e \in \mathcal{E}$ to a collection of sets, each set containing a fragment of K (axioms) that cause the event to occur. This tracing function is maintained throughout the application of tableau expansion rules defined in [6].

For purpose of this work, the original set of *change events* has been extended [1] to include all possible events that can occur during the application of expansion rules. The extension of events includes the following events:

- **Add** (x, \mathcal{V}) represents the action of adding a node x to the vertex set, \mathcal{V} , a completion graph, i.e. the operation $\mathcal{V} \leftarrow \mathcal{V} \cup \{x\}$.
- **Add** $(\langle x, y \rangle, \mathcal{E})$ represents the action of adding a edge $\langle x, y \rangle$ to the edge set, \mathcal{E} , in a completion graph, i.e. the operation $\mathcal{E} \leftarrow \mathcal{E} \cup \{\langle x, y \rangle\}$.
- **Remove** (x, \mathcal{V}) represents the action of removing a node x from the vertex set, \mathcal{V} , a completion graph, i.e. the operation $\mathcal{V} \leftarrow \mathcal{V} \setminus \{x\}$.
- **Remove** $(\langle x, y \rangle, \mathcal{E})$ represents the action of removing an edge $\langle x, y \rangle$ from the edge set, \mathcal{E} , in a completion graph, i.e. the operation $\mathcal{E} \leftarrow \mathcal{E} \setminus \{\langle x, y \rangle\}$.
- **Remove** $(C, \mathcal{L}(x))$ represents the action of removal a concept C from the label of a node x , i.e. the operation $\mathcal{L}(x) \leftarrow \mathcal{L}(x) \setminus \{C\}$.
- **Remove** $(R, \mathcal{L}(\langle x, y \rangle))$ represents the removal of a role R from the label of an edge $\langle x, y \rangle$, i.e., $\mathcal{L}(\langle x, y \rangle) \leftarrow \mathcal{L}(\langle x, y \rangle) \setminus \{R\}$.
- **Remove** $x \neq y$ represents the removal of an inequality relation between two nodes x, y , where x, y , i.e. $\neq \leftarrow \neq \setminus \{\langle x, y \rangle\}$.
- **Remove** $x = y$ represents the removal of an inequality relation between two nodes x, y .

Additionally the tracing function maintenance through expansion rule application has been extended. Although the tracing extension has been provided for *SHOIQ*, it is trivial to see how they are applied to *SHIQ* [5] and *SHOQ* [4] completion strategies. Further details are omitted, however they can be found in [1].

In general, the update algorithm for deletion works as follows. When an ABox axiom is removed, the algorithm performs a lookup in the graph for all *change events* whose axiom traces include the axiom number of the deleted axiom. These events are *rolled-back* if and only if their axiom trace sets only include sets which contain the deleted axiom, possibly among other axioms. By *roll-back* we refer to simply undoing (the inverse) the event (e.g., rolling back the event $Add(x, \mathcal{V})$ would be the process of removing x from \mathcal{V}). If there exists additional axiom traces for that particular event that do not include the removed axiom, then only the sets including the removed axiom are deleted from the axiom trace set; in this situation the actual event is not *rolled-back*. This is intuitive as there still exists support for that particular event.

As in the approach for additions, the completion rules must be reapplied the update completion graph as it is possible for the graph to be incomplete for the KB. One additional requirement is that if a clash occurs after a removal and backjumping is performed, all possible branches must be explored. That is, branches in which there a clash prior to a deletion may need to be re-explored as the clash may no longer exist after an axiom was removed.

Axiom tracing additionally requires a slight modification to the update approach for ABox additions. For example, in the case that a individual type assertion is added to the KB, the algorithm must add a new tracing set to the axiom trace for the affected components (this set will consist of the new axiom number). The update algorithm ($UPDATE(G, \alpha)$) is provided in Figure 1. Note that *deps* (dependents) is the inverted tracing function index. Additionally, the operator \bowtie is defined as follows: let $\mathbf{S}_1 = \{S_1^1, \dots, S_m^1\}$ and $\mathbf{S}_2 = \{S_1^2, \dots, S_n^2\}$ be two collections of sets, then:

$$\begin{aligned} S_i^1 (1 \leq i \leq m) \in S_1 \bowtie S_2 &\text{ iff } S_i^1 \not\subset S_j^2 \text{ for all } j, 1 \leq j \leq n \\ S_j^2 (1 \leq j \leq n) \in S_1 \bowtie S_2 &\text{ iff } S_j^2 \not\subset S_i^1 \text{ for all } i, 1 \leq i \leq m \end{aligned}$$

We now provide the correctness proof sketch for update algorithm under ABox additions.

Theorem 1 *Let \mathbf{K} be a knowledge base, α be an added ABox assertion, $CR()$ be the tableau expansion rules function (as defined previously), and let G be a tableau completion graph such that:*

$$\{G \mid G \in CR(\mathbf{K})\}$$

Then, $UPDATE(G, \alpha) \in \{G' \mid G' \in (CR(\mathbf{K} \cup \{\alpha\}))\}$

Proof Sketch

Due to the non-determinism of the application of completion rules, it is obvious that every graph generated from a subpart of a KB is a possible state of the completion graph for the KB, which is potentially incomplete as further rules can fire. Given the completion graph G for \mathbf{K} , it can therefore be shown that G can be arrived at by firing some subset of the completion rules on $\mathbf{K} \cup \{\alpha\}$. Because

```

function UPDATE(  $G, \alpha$  )
  if  $\alpha$  is an addition then
    if  $\alpha$  is a  $x \neq y$  or  $x = y$  then
      let  $op$  be the operation, such that  $op \in \{=, \neq\}$ 
      if  $x \notin \mathcal{V}$  then
         $\mathcal{V} \leftarrow \mathcal{V} \cup \{x\}$ 
      if  $y \notin \mathcal{V}$  then
         $\mathcal{V} \leftarrow \mathcal{V} \cup \{y\}$ 
      if  $op$  is  $\neq$  then
        if  $x \neq y \in x \neq y$  then add it
      if  $op$  is  $=$  then
        Merge  $x$  and  $y$ 
         $\tau(x \text{ op } y) \leftarrow \tau(x \text{ op } y) \uplus \{\{\alpha\}\}$ 
         $\tau(\text{Add}(x, \mathcal{V})) \leftarrow \tau(\text{Add}(x, \mathcal{V})) \uplus \{\{\alpha\}\}$ 
         $\tau(\text{Add}(y, \mathcal{V})) \leftarrow \tau(\text{Add}(y, \mathcal{V})) \uplus \{\{\alpha\}\}$ 
         $\text{deps}(\alpha) \leftarrow \text{deps}(\alpha) \cup \{\{x \text{ op } y\}\{\text{Add}(x, \mathcal{V})\}\{\text{Add}(x, \mathcal{V})\}\}$ 
      else if  $\alpha$  is a individual type addition,  $(C(x))$  then
        if  $x \notin \mathcal{V}$  then
           $\mathcal{V} \leftarrow \mathcal{V} \cup \{x\}$ 
           $\mathcal{L}(x) \leftarrow \mathcal{L}(x) \cup \{C\}$ 
           $\tau(\text{Add}(x, \mathcal{V})) \leftarrow \tau(\text{Add}(x, \mathcal{V})) \uplus \{\{\alpha\}\}$ 
           $\tau(\text{Add}(C, \mathcal{L}(x))) \leftarrow \tau(\text{Add}(C, \mathcal{L}(x))) \uplus \{\{\alpha\}\}$ 
           $\text{deps}(\alpha) \leftarrow \text{deps}(\alpha) \cup \{\{\text{Add}(x, \mathcal{V})\}\{\text{Add}(C, \mathcal{L}(x))\}\}$ 
        else if  $\alpha$  is a role assertion addition,  $R(x,y)$  then
          if  $\langle x, y \rangle \notin \mathcal{E}$  then
             $\mathcal{E} \leftarrow \mathcal{E} \cup \{\langle x, y \rangle\}$ 
             $\mathcal{L}(\langle x, y \rangle) \leftarrow \mathcal{L}(e) \cup \{R\}$ 
             $\tau(\text{Add}(x, \mathcal{V})) \leftarrow \tau(\text{Add}(x, \mathcal{V})) \uplus \{\{\alpha\}\}$ 
             $\tau(\text{Add}(y, \mathcal{V})) \leftarrow \tau(\text{Add}(y, \mathcal{V})) \uplus \{\{\alpha\}\}$ 
             $\tau(\text{Add}(\langle x, y \rangle, \mathcal{E})) \leftarrow \tau(\text{Add}(\langle x, y \rangle, \mathcal{E})) \uplus \{\{\alpha\}\}$ 
             $\tau(\text{Add}(R, \mathcal{L}(\langle x, y \rangle))) \leftarrow \tau(\text{Add}(R, \mathcal{L}(\langle x, y \rangle))) \uplus \{\{\alpha\}\}$ 
             $\text{deps}(\alpha) \leftarrow \text{deps}(\alpha) \cup \{\{\text{Add}(x, \mathcal{V})\}\{\text{Add}(y, \mathcal{V})\}\{\text{Add}(\langle x, y \rangle, \mathcal{E})\}\{\text{Add}(R, \mathcal{L}(\langle x, y \rangle))\}\}$ 
          Apply expansion rules to  $G$ 
          if there is a clash then
            Perform backjumping
          else if  $\alpha$  is a deletion then
             $\text{events} \leftarrow \text{deps}(\alpha)$ 
             $\text{deps}(\alpha) \leftarrow \emptyset$ 
            for each  $e \in \text{events}$  do
               $\text{traces} \leftarrow \tau(e)$ 
              for each  $t \in \text{traces}$  do
                if  $\alpha \in t$  do
                   $\text{traces} \leftarrow \text{traces} \setminus t$ 
                if  $\text{traces} = \emptyset$  do
                  roll-back  $e$ 
                   $\tau(e) \leftarrow \text{traces}$ 
            Apply expansion rules to  $G$ 
            if there is a clash then
              Perform backjumping
          return  $G$ 

```

Figure 1: Pseudo-code of update procedure for $SHOQ(\mathcal{D})$ and $SHIQ(\mathcal{D})$ ABoxes

completion graph can be incomplete related to $K \cup \{\alpha\}$, the rule firing for α must then be applied. It can therefore be shown that $UPDATE(G, \alpha) \in \{G' | G' \in (CR(K \cup \{\alpha\}))\}$

□

Additionally, the correctness proof for the update algorithm under ABox deletions is presented.

Theorem 2 *Let K be a knowledge base, α be an deleted ABox assertion, $CR()$ be the tableau expansion rules function (as defined previously), and let G be a tableau completion graph such that:*

$$\{G | G \in CR(K)\}$$

Then, $UPDATE(G, \alpha) \in \{G' | G' \in (CR(K - \{\alpha\}))\}$

Proof Sketch

The tracing function captures all actions in G that were caused in part by α . It can be shown that by removing all structures (nodes/edges/labels) that are *only* reliant on an axiom trace that includes α , G is effectively *rolled-back* to a state in which the effects of the rule firings caused by α are removed. Because $UPDATE(G, \alpha)$ performs this rollback, it can be shown that a completion graph is obtained which is a possible state of the a completion graph for the KB. While this graph is potentially incomplete, reapplying expansion rules guarantees the algorithm will arrive at some graph that could be obtained by simply applying the completion rules to $K \cup \{\alpha\}$. It can therefore be shown that $UPDATE(G, \alpha) \in \{G' | G' \in (CR(K - \{\alpha\}))\}$

□

4 Implementation and Evaluation

We have implemented the approach presented in this paper in an open source OWL-DL reasoner, Pellet ². In order to evaluate our update algorithm, we have performed an emperical evaluation using two different KBs with large ABoxes - the Lehigh University Benchmark (LUBM)³ and AKT Reference Ontology ⁴.

Three tests were run over three KBs consisting of one, two, and lastly three universities from the LUBM dataset generator. First an initial consistency check was performed and then in each test, a random update was selected which was used to update the KB. In the regular version of the reasoner, the cached completion graph was discarded, while in the optimized reasoner the update algorithm was utilized. For each KB size, varying sized additions and deletions were randomly selected from the

²Pellet OWL-DL Reasoner: <http://www.mindswap.org/2003/pellet/>

³LUBM Ontology: <http://swat.cse.lehigh.edu/projects/lubm/>

⁴AKT Ontology: <http://www.aktors.org/publications/ontology/>

dataset. Update sizes include single axiom, twenty-five axioms, and fifty axioms (individual and/or role). Each test was performed twenty-five times and the results were averaged. Expressivity and KB statistics are provided in Table 1.

Name	Classes / Properties / Individuals / Assertions	Expressivity
LUBM-1 Univ	43 / 32 / 18,257 / 97,281	<i>SHI</i>
LUBM-2 Univ	43 / 32 / 47,896 / 254,860	<i>SHI</i>
LUBM-3 Univ	43 / 32 / 55,110 / 295,728	<i>SHI</i>
AKT-1	160 / 152 / 16,505 / 70,948	<i>SHIF</i>
AKT-2	160 / 152 / 32,926 / 143,334	<i>SHIF</i>

Table 1: LUBM and AKT Portal Dataset Statistics

Results for additions and removals in the LUBM test are presented in Figure 2 (timing results are shown in milliseconds and the scale is logarithmic). We note that the '0' axiom value represents the initial consistency check. In both versions

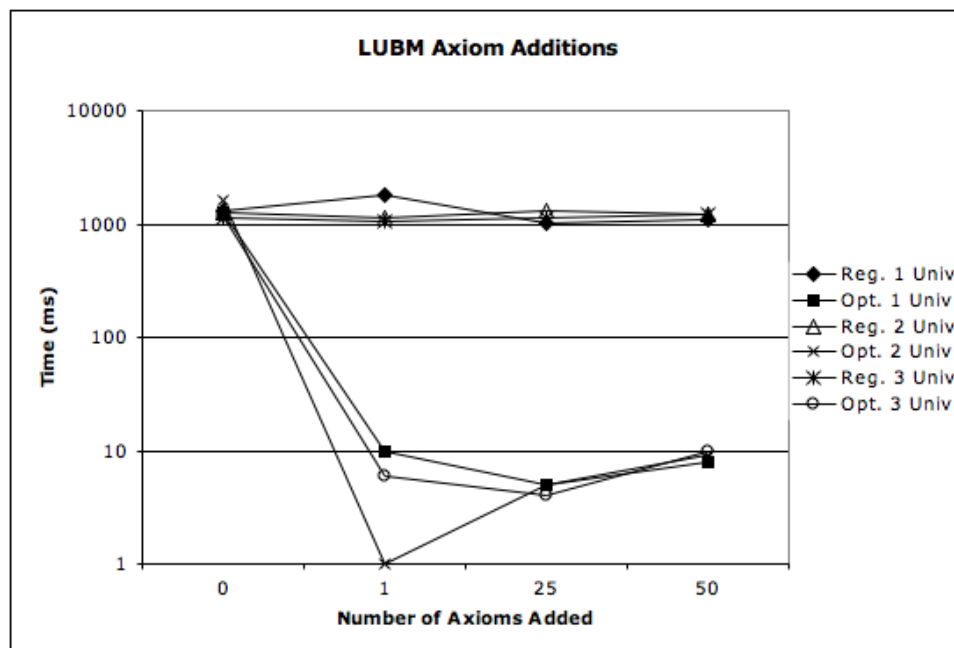


Figure 2: Addition Updates of LUBM datasets

of the reasoner, initial consistency checks are comparable. However for both update types, performance improvements ranging from one to three orders of magnitude are achieved under updates in the reasoner with the optimized update algorithm. This is due to the avoidance of re-firing of completion rules by maintaining the previous completion graph; therefore very few (if any in some cases) completion rules must be refired. This provides direct empirical evidence for the effectiveness of the update

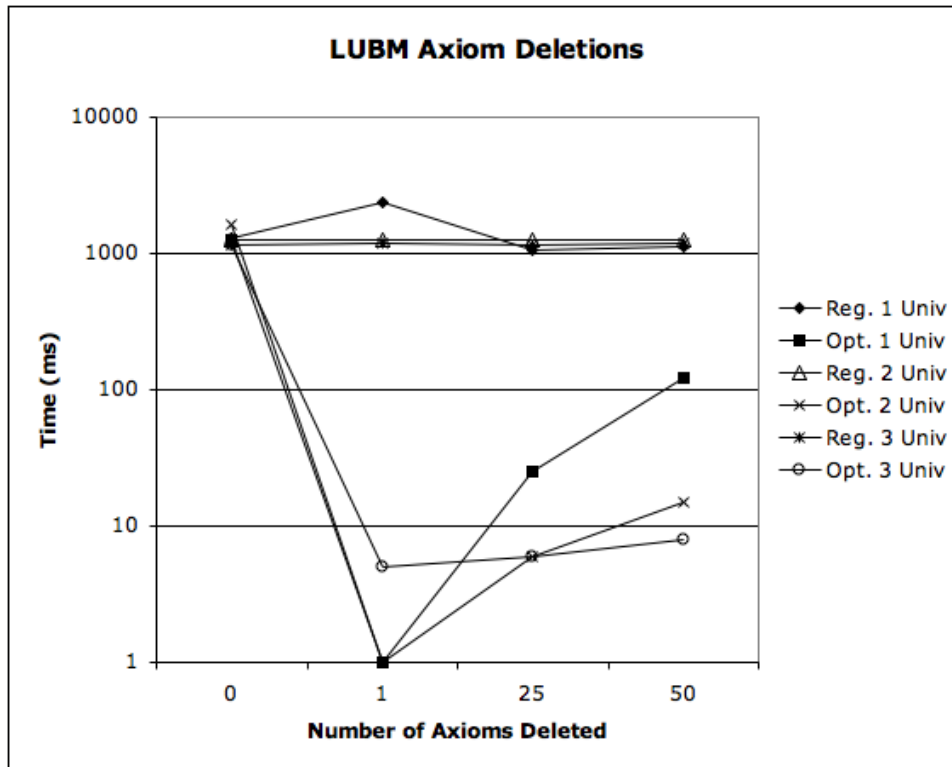


Figure 3: Deletion Updates of LUBM datasets

algorithm. In a second evaluation, two datasets⁵ adhering to the AKT Reference ontology were used (statistics shown in Table 1). The tests were structured in the same manner as the LUBM test. Again, each test was performed twenty-five times and the results are averaged over these iterations. All timings are in milliseconds and the scale is logarithmic. Similar to the LUBM test, update performance is improved between one to three orders of magnitude. Note that the performance of the update is better in the LUBM test cases; this is primarily due to the increased complexity of the AKT Reference Ontology. Therefore, a larger number of expansion rules are applied after the update. However the update algorithm greatly outperforms the regular reasoner, again demonstrating the effectiveness and overall impact of the update algorithm.

5 Discussion and Future Work

We are currently working on extending this work for the Description Logics *SHOIQ*. However, we note that the update procedure is complicated due to the completion rule

⁵Hyphen-REA:
http://www.hyphen.info/rdf/hero_complete.zip

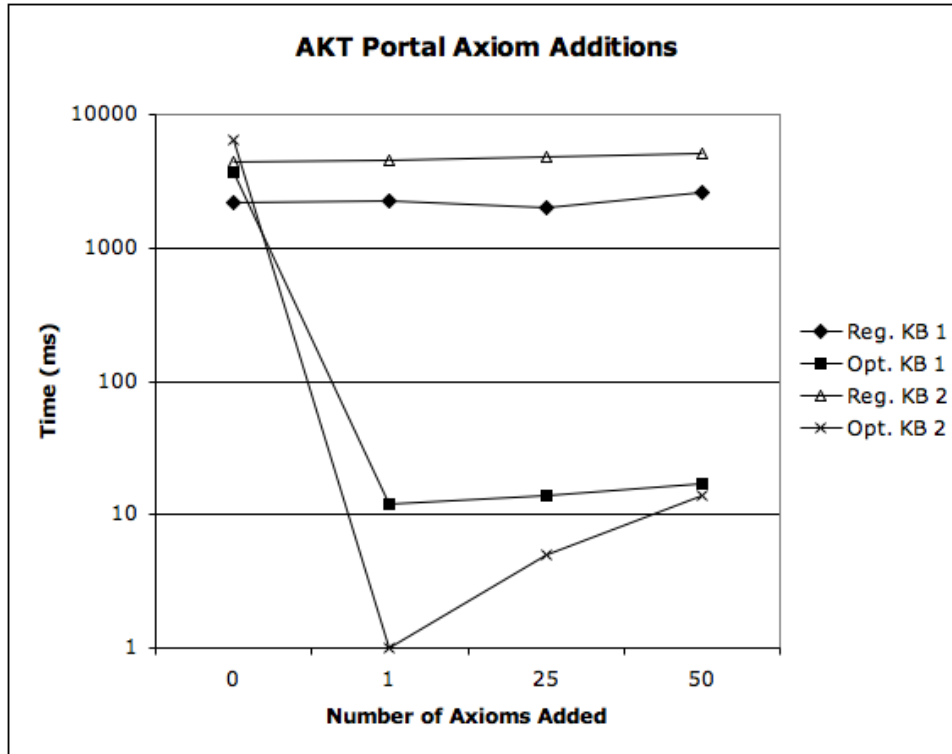


Figure 4: Addition Updates of AKT datasets

application ordering which is imposed in [6].

The algorithm presented in this paper only provides update for axiom additions and removals. However, one may wish to provide support for axiom revision. However, this can be reduced to a deletion and directly followed by an addition. We leave the investigation of a more fine-grained update algorithm as future work.

While we achieved dramatic results for consistency checking under ABox updates, one may wish to update TBox axioms as well. This presents the additional issue of rolling back through the pre-processing steps such as absorption. We are currently investigating this problem. This work only directly addresses optimizing consistency checking under ABox updates; however standard reasoning tasks in Description Logic reasoners, including classification, realization, and query answering are all reducible to KB consistency checking [2]. Hence, we would like to investigate the utility of the update algorithm for generalized reasoning services.

6 Related Work

To our knowledge there has been no previous work in DL reasoning algorithms for incremental maintenance of completion graphs. We do note that this work can be

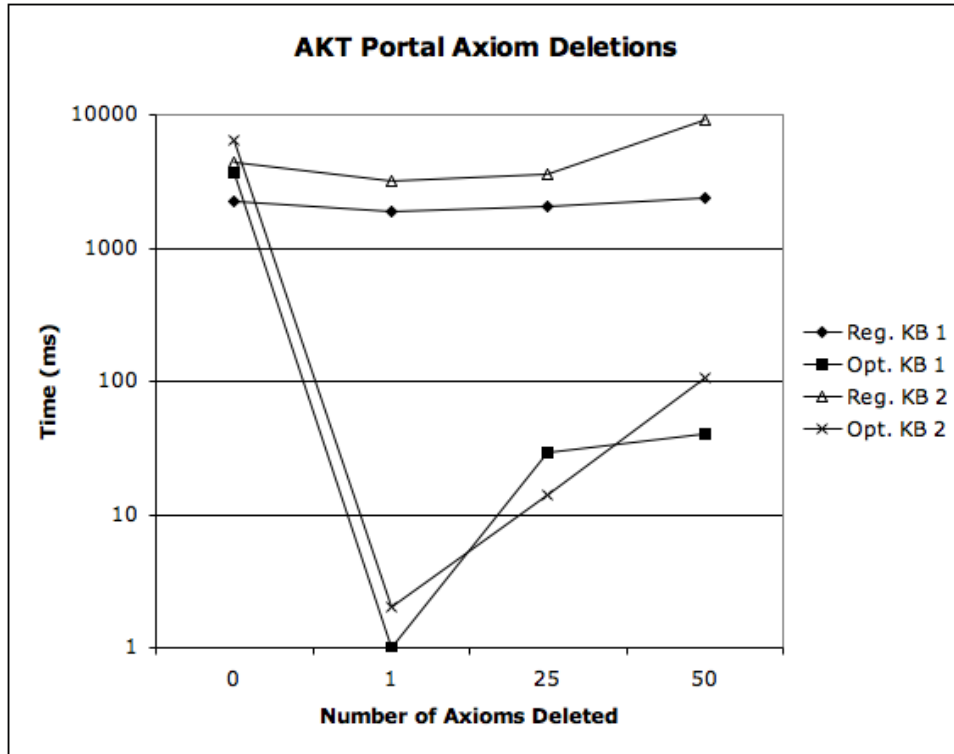


Figure 5: Deletion Updates of AKT datasets

paralleled to view maintenance and truth maintenance; however we deal with a more expressive logic and a different proof theory.

There has also been recent work in optimizing incremental instance retrieval in expressive description logics [3]; however previous work has not addressed the notion of updating the completion graphs for consistency checking.

Additionally, in [9, 10] the authors specify update semantics for descriptions logics; however this work is less related to belief revision and is independent as we are concerned with maintaining the internal state of the reasoner.

7 Conclusion

Modern description logic reasoners are traditionally oriented toward providing reasoning services for relatively static ontologies. However, there are numerous situations in which the ontology itself is in flux, requiring frequent updates. This includes Web portals, sensor networks, and ontologies editors.

In this paper, we have presented an algorithm for updating completion graphs for the Description Logics $SHIQ(\mathcal{D})$ and $SHOQ(\mathcal{D})$ under both the addition and removal of ABox assertions, providing a critical step towards reasoning procedures

for fluctuating or streaming data. We have provided an empirical analysis of the algorithm through an experimental implementation in the Pellet reasoner. Our initial results are very promising as they demonstrate orders of magnitude performance improvement.

References

- [1] F. Baader and W. Nutt. Basic description logics. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.
- [2] Volker Haarslev and Ralf Moller. Incremental query answering for implementing document retrieval services. In *International Workshop on Description Logics*, pages 85–94, 2003.
- [3] Christian Halaschek-Wiener, Aditya Kalyanpur, and Bijan Parsia. Extending tableau tracing for abox updates. In *UMIACS Technical Report*, 2006. <http://www.mindswap.org/papers/2006/aboxTracingTR2006.pdf>.
- [4] I. Horrocks and U. Sattler. Ontology reasoning in the SHOQ(D) description logic. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, 2001.
- [5] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of the 6th Int. Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.
- [6] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for SHOIQ. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. Morgan Kaufman, 2005.
- [7] Aditya Kalyanpur, Bijan Parsia, Bernardo Cuenca-Grau, and Evren Sirin. Tableau tracing in shoin. UMIACS Tech Report.
- [8] Hirofumi Katsuno and Alberto Mendelzon. On the difference between updating a knowledge base and revising it. In *International Conference of Principles of Knowledge Representation and Reasoning(KR)*, 1991.
- [9] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic aboxes. In *International Conference of Principles of Knowledge Representation and Reasoning(KR)*, 2006.

- [10] Mathieu Roger, Ana Simonet, and Michel Simonet. Toward updates in description logics. In *International Workshop on Knowledge Representation meets Databases*, 2002.
- [11] M. Winslett. Updating logical databases. In *Updating Logical Databases*. Cambridge University Press, 1990.