Description Logic Reasoning with Syntactic Updates

Christian Halashek-Wiener[1], Bijan Parsia[2], and Evren Sirin[1]

[1] Department of Computer Science,
University of Maryland, College Park, MD USA
{halasche, sirin}@cs.umd.edu
[2] School of Computer Science,
The University of Manchester, UK
bparsia@cs.man.ac.uk

Abstract. Various data sources on the Web tend to be highly dynamic; this is evident in prominent Web services frameworks in which devices register or deregister their descriptions quite rapidly and in Semantic Web portals which allow content authors to modify or extend underlying ontologies and submit content. Such applications often leverage Description Logic (DL) reasoning for a variety of tasks (e.g., classifying Web service descriptions, etc); however, this can introduce substantial overhead due to content fluctuation, as DL reasoners have only been considered for relatively static knowledge bases. This work aims to provide more efficient DL reasoning techniques for frequently changing instance bases (ABoxes). More specifically, we investigate the process of incrementally updating tableau completion graphs used for reasoning in the expressive DLs $\mathcal{SHOQ}$ and $\mathcal{SHIQ}$, which correspond to a large subset of the W3C standard Web Ontology Language, OWL-DL. We present an algorithm for updating completion graphs under the syntactic addition and removal of ABox assertions. We also provide an empirical analysis of the approach through an implementation in the OWL-DL reasoner, Pellet.

1 Introduction

Recently, there has been increased interest in providing formal representation of Web content using ontologies that correspond to expressive Description Logics (DLs). Due to data sources that produce fluctuating data, there exists a variety of reasoning use cases which require frequent updates at both the assertional (ABox) and terminological (TBox) levels, some of which are briefly highlighted below:

- Prominent web services frameworks (e.g., OWL-S) use Description Logics for service discovery and matchmaking [24,25,21]. Services, especially device services in pervasive contexts, may register or deregister their descriptions (and supporting ontologies) quite rapidly, yet the matchmaking service must remain responsive.
- Semantic Web portals often allow content authors to modify or extend the ontologies which organize their site structure or page content. While in some cases a "defer update" strategy is acceptable, in general it is more gratifying to users if changes they make are reflected immediately in the site.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 722–737, 2006.

- Perhaps the single most common use of Description Logic reasoners is in ontology editors. Most editors do not do continuous reasoning while one is editing (one exception is [18]), relying on an analogue of the edit-compile-test loop of most programming environments. However, if this cycle is very long (e.g., hundreds of seconds) then users will be forced to perform larger sets of edits before testing. This discourages experimentation, particularly in debugging contexts.
- Syndication (publish/subscribe) systems on the Web have recently been transitioning to more expressive approaches; that is subscribers (and publishers) are provided with more expressive means for describing their interests (resp. published content), enabling more accurate dissemination. Recently, there has been interest in utilizing OWL and DL reasoning for the purpose of matching published content with subscription requests [8,9,26,17]. When disseminating time sensitive information (e.g., in the financial domain), efficient reasoning for frequently changing KBs (due to continuously published information) will be critical in achieving practical DL-based approaches.

While there exists such use cases for reasoning under changing data, current DL reasoning algorithms have been developed considering relatively static knowledge bases. In this paper, we investigate performing incremental consistency checking in the expressive Description Logics $\mathcal{SHOQ}$ and $\mathcal{SHIQ}$, which correspond to a large subset of the W3C standard Web Ontology Language, OWL-DL. In particular, we present an approach for incrementally updating tableau completion graphs created during consistency checks under syntactic addition and removal of ABox (instance) assertions. This provides a critical step towards DL reasoning over fluctuating data, as it has been shown that in KBs with substantially sized ABoxes, consistency checking can dominate reasoning time; further, all standard reasoning tasks are reduced to consistency checks. Lastly, we provide an empirical analysis of the optimizations through an experimental implementation in an open source OWL-DL reasoner, Pellet.

2 Preliminaries

In this section, we briefly discuss background information directly relevant to this work. First, we present the syntax and semantics of the Description Logic $\mathcal{SHOIQ}$, which corresponds to OWL-DL, with the slight extension of qualified cardinality restrictions. Additionally, we provide a brief overview of tableau algorithms for Description Logic reasoning.

2.1 $\mathcal{SHOIQ}$ Description Logic

Let N_C, N_R, N_I be non-empty and pair-wise disjoint sets of *atomic concepts*, *atomic roles* and *individuals* respectively. The set of $\mathcal{SHOIQ}$ roles (roles, for short) is the set $N_R \cup \{R^- | R \in N_R\}$, where R^- denotes the inverse of the atomic role R. Concepts are inductively defined using the following grammar:

$$C \leftarrow A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \bowtie nS.C \mid \{a\}$$

where $A \in N_C$, $a \in N_I$, $C_{(i)}$ a $\mathcal{SHOIQ}$ concept, R a role, S a *simple* role [1] and $\bowtie \in \{\leq, \geq\}$. We write $\top$ and $\bot$ to abbreviate $C \sqcup \neg C$ and $C \sqcap \neg C$ respectively.

A *role inclusion axiom* is an expression of the form $R_1 \sqsubseteq R_2$, where R_1, R_2 are roles. A *transitivity axiom* is an expression of the form $Trans(R)$, where $R \in V_R$. An RBox R is a finite set of role inclusion axioms and transitivity axioms. For C, D concepts, a *concept inclusion axiom* is an expression of the form $C \sqsubseteq D$. A TBox T is a finite set of concept inclusion axioms. An ABox A is a finite set of concept assertions of the form $C(a)$ (where C can be an arbitrary concept expression) and role assertions of the form $R(a, b)$. A *knowledge base* $\mathsf{K} = (\mathsf{T}, \mathsf{R})$ consists of a TBox and an RBox.

An *interpretation* $\mathcal{I}$ is a pair $\mathcal{I} = (\mathcal{W}, .^{\mathcal{I}})$, where $\mathcal{W}$ is a non-empty set, called the *domain* of the interpretation, and $.^{\mathcal{I}}$ is the *interpretation function*. The interpretation function assigns to $A \in N_C$ a subset of $\mathcal{W}$, to each $R \in N_R$ a subset of of $\mathcal{W} \times \mathcal{W}$ and to each $a \in N_I$ an element of $\mathcal{W}$. The interpretation function is extended to complex roles and concepts as given in [14].

The satisfaction of a $\mathcal{SHOIQ}$ axiom α in an interpretation $\mathcal{I}$, denoted $\mathcal{I} \models \alpha$ is defined as follows: (1) $\mathcal{I} \models R_1 \sqsubseteq R_2$ iff $(R_1)^{\mathcal{I}} \subseteq (R_2)^{\mathcal{I}}$; (2) $\mathcal{I} \models Trans(R)$ iff for every $a, b, c \in \mathcal{W}$, if $(a, b) \in R^{\mathcal{I}}$ and $(b, c) \in R^{\mathcal{I}}$, then $(a, c) \in R^{\mathcal{I}}$; (3) $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; The interpretation $\mathcal{I}$ is a model of the RBox R (respectively of the TBox T) if it satisfies all the axioms in R (respectively T). $\mathcal{I}$ is a model of $\mathsf{K} = (\mathsf{T}, \mathsf{R})$, denoted by $\mathcal{I} \models \mathsf{K}$, iff $\mathcal{I}$ is a model of T and R.

2.2 Tableau Algorithms

The algorithm presented here is based on the tableau decision procedure for ABox consistency checking in $\mathcal{SHOQ}$ [12] and $\mathcal{SHIQ}$ [13]. DL tableau-based algorithms decide the consistency of an ABox A w.r.t a TBox T (by TBox, we are additionally referring to all axioms for roles) by trying to construct (an abstraction of) a common model for A and T, called a *completion graph* [14]. Formally, a completion graph for an ABox A with respect to T is a directed graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \dot{\neq})$. 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 $\dot{\neq}$ is used for recording inequalities between nodes.

The completion graph is constructed by repeatedly applying a set of *expansion rules*. Whenever a contradiction is encountered, a DL reasoner will either backtrack and select a different non-deterministic choice, or report the inconsistency and terminate if no choice remains to be explored. While there may exist more than one model for A and T, the tableau algorithm will only find one such model. We also point out that *blocking* is utilized to ensure termination of tableau algorithms [11]. Blocking is used to detect cycles encountered during the application of tableau expansion rules, therefore stopping the expansion of a branch when the same labels reoccur. For example, blocking is clearly necessary to ensure termination for the following KB, $\mathsf{K} = \{a{:}C, C \sqsubseteq \exists R.C\}$. Further details can be found in [11]. We lastly note that in our work, we utilize a slightly modified version of the $\mathcal{SHOQ}$ and $\mathcal{SHIQ}$ expansion rules, which are augmented for axiom tracing as defined in [10,16,15].

[1] See [14] for a precise definition of simple roles.

3 Syntactic ABox Updates

In this work, we consider ABox addition and deletion of individual equality and inequality assertions $x = y$ and $x \neq y$, concept assertions $x{:}C$ and role assertions $\langle x, y\rangle{:}R$. In general updating ABox assertions in the presence of a TBox and an RBox brings up several issues with the semantics.

For the purpose of this work, we adopt syntactic changes/updates of ABox assertions, which we refer to as *Syntactic Updates*. By *syntactic* we refer to the explicitly asserted ABox facts; this is similiar to the distiction between belief bases and belief sets in belief revision literature [20]. Intuitively, *Syntactic Updates* can be described as an update in which all new ABox assertions are directly added (or removed) to the asserted (base) axioms; therefore the only changes that occur are those explicitly stated in the ABox update. Removing an assertion from the ABox under these semantics does not guarantee that the removed assertion will not be entailed anymore. Furthermore, in this work we do not address resolving inconsistencies introduced by the addition of a new axiom. Formally, we describe this update as follows:

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

This type of ABox update is different when compared to related work in formal update semantics [19,23] and belief revision [5,6] for DLs, however real world use of this type of changes is directly present in many document-oriented services, including Semantic Web Service repositories, publish/subscribe (syndication) application, Semantic Web portals, etc. 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 *Syntactic Updates*; therefore we feel these semantics is warranted. Lastly, we note that ABox additions under *Syntatic Updates* is similar to the *expansion* operator as traditionally defined belief revision [1].

4 Update Approach

The goal of the approach presented here is to update a previously constructed completion graph in such a way that it is guaranteed to correspond to a model of the updated KB (in the event some model exists). The approach presented here is applicable to the Description Logics $\mathcal{SHOQ}$ [12] and $\mathcal{SHIQ}$ [13], as the difference between the two completion algorithms is independent of the update algorithm.

4.1 ABox Additions

Conceptually, tableau algorithms for $\mathcal{SHOQ}$ [12] and $\mathcal{SHIQ}$ [13] can be thought of as incremental in that *expansion rules* are repeatedly 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, $x{:}C$, the approach 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 occur in $\mathcal{L}(x)$. If an individual inequality relation, $x \neq y$, is added to the KB, the algorithm checks if $x \in \mathcal{V}$ and $y \in \mathcal{V}$. If either does not exists, then they are added to the graph. Additionally, if the $\langle x, y\rangle \notin x \dot{\neq} y$, then it is added. Alternatively, if an individual equality relation, $x = y$, is added to the KB, the approach checks if $x \in \mathcal{V}$ and $y \in \mathcal{V}$. If either does not exists, then they are added to the graph. Additionally, x and y are merged. Lastly, if a role, $\langle x, y\rangle{:}R$, is added to the KB 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 $R \notin \mathcal{L}(\langle x, y\rangle)$, then R is added to $\mathcal{L}(\langle x, y\rangle)$.

After the graph has been updated, the completion rules must be reapplied to the updated completion graph, as the update may cause 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.

4.2 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 [2,15,16], which tracks the dependencies of completion 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*, denoted EV, that change the state of the completion graph, or the flow of the algorithm. In [15,16] a set of change events is defined, which include the following:

- 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 E y is the action of *merging* the nodes x, y. A detailed description of the *merge* operation can be found in [14].
- x NE y stands for the addition of an inequality relation between two nodes x,y i.e. $\dot{\neq} \leftarrow \dot{\neq} \cup \{\langle x, y\rangle\}$.

In order to record the changes on the completion graph, [15,16] introduces a *tracing function*, which keeps track of the asserted axioms responsible for changes to occur. The *tracing function*, τ, maps each event $e \in EV$ to a set of sets, each set containing

a fragment of the KB that cause the event to occur. This tracing function is maintained throughout the application of tableau expansion rules.

For purpose of this work, the original set of *change events* has been extended [10] to include all possible events that can occur during the application of expansion rules. The extension of events includes the following additional *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 E y represents the action of undoing a previous *merge* between the nodes x, y. A detailed description of the *merge* operation can be found in [14].
- Remove x $\dot{NE}$ y represents the removal of an inequality relation between two nodes x,y, i.e. $\dot{\neq} \leftarrow \dot{\neq} \setminus \{\langle x, y\rangle\}$

Additionally, the tracing function maintenance through expansion rule application has been extended [10]. Although the tracing extension has been provided for $\mathcal{SHOIQ}$ [14], it is trivial to see how they are applied to $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ completion strategies. Further details can be found in [10].

The general idea is to utilize axiom tracing in order to track the dependencies of parts of the completion graph, so that the effects of ABox assertions being removed can be *rolled-back*. We note here that portions of the completion graph can be introduced by multiple (explicitly) asserted axioms, however each concept or role name in a label will only occur once. Therefore, axiom traces must be maintained for *each* asserted axiom sets that can potentially introduce a particular concept or role name in a label.

To clarify, consider the following KB, $\mathsf{K} = \{1.\ a{:}C \sqcap D,\ 2.\ D \sqsubseteq B\}$ (axiom numbers are provided for tracing purposes). After the completion graph is initially created for this KB, the node in the graph corresponding to a would have a variety of concept names in $\mathcal{L}(a)$, including D with an axiom trace of $\{1\}$ (this would be added from the $\sqcap$-rule) and B with an axiom trace of $\{1, 2\}$ (this would be added from the $unfolding$-rule). Next consider an incremental update of (Add $a{:}D$); since there already exists a trace for D, another axiom set is added to its trace. Therefore the axiom traces are sets of traces, as defined in [10,15,16]. In this case the axiom trace for $D \in \mathcal{L}(a)$ would be $\{\{1\}, \{3\}\}$, where $\{3\}$ corresponds to the added assertoin. We note here that in [15,16], the tableau algorithm runs to saturation (i.e, it continues applying all expansion rules until no rules are applicable, even if a clash is detected). However for purpose of this work, we use the normal tableau termination procedure, in which the algorithm proceeds until either

all completion graphs are closed or one complete and clash-free completion graph is found. We additionally note that this does not affect the tracing procedure.

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 holds as there still exists support for that particular event.

As in the approach for additions, the completion rules must be reapplied to the updated completion graph as it is possible for the graph to be incomplete. Axiom tracing additionally requires a slight modification to the update approach for ABox additions in order to maintain axiom traces. 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).

4.3 Update Algorithm

We now define the update algorithm $UPDATE(G, \alpha)$, which takes as input a completion graph G (for a knowledge base K) and an update α and returns a new completion graph; the algorithm is shown in Figure 1. Note that τ is the tracing function and $deps$ (dependents) is the inverted tracing function index (asserted axiom to a set of change events).

Now we provide the correctness proof for update algorithm under ABox additions. We first make the following observation: due to the *generating* tableau expansion rules, namely the $\exists - rule$ and the $\geq -rule$, new individuals could have been introduced to the graph that would not have been added in the completion graph if it were built from scratch; therefore it cannot simply be shown that $UPDATE(G, \alpha)$ will obtain a completion graph that could have been built if we applied the expansion rules to the updated KB (explicit ABox and TBox assertions). For example, this is evident if there is an addition $b{:}C$ to an ABox that consists of $a{:}\exists R.C$ and $\langle a, b\rangle{:}R$. If the completion graph where built from scratch, the $\exists$-rule would be blocked for the node with label $\exists R.C$ as there already exists a R-neighbor b (i.e., $R \in \mathcal{L}(b)$). In contrast, in $UPDATE(G, \alpha)$ the $\exists - rule$ would have already fired prior to the addition.

Theorem 1. *Under ABox additions, $UPDATE(G, \alpha)$ is sound, complete, and terminating.*

Proof Sketch

It is obvious that every graph generated from a subpart of a KB is a possible state of the completion graph for the KB, though it is potentially incomplete as more rules can fire. In $UPDATE(G, \alpha)$, the newly induced structure from the syntactic update is added to the graph, as it would've existed if the completion graph for $\mathsf{K} \cup \{\alpha\}$ were built from

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

Fig. 1. Pseudo-code of update procedure for $\mathcal{SHOQ}$ and $\mathcal{SHIQ}$ KBs

scratch. From our discussion earlier, it is clear the initially updated completion graph can contain clashes; however the completion rules are then re-fired. It can therefore be shown that the resulting completion graph must correspond to a model if and only if at least one exists, as if it were the case that the completion graph did not correspond to a model (via some clash), then the soundness or completeness of [12,13] would be contradicted (i.e., if a model exists yet the initially updated completion graph contains a clash, it would be removed by shrinking expansion rules or backjumping). Further, if a non-deterministic choice were taken and backjumping occurs, the newly added

structures imposed by the update will remain, as they are explicitly asserted. It is clear that if some model exists, the tableau algorithm then proceeds as usual and a closed, clash-free completion graph is found. Therefore, it can be shown that under ABox additions, $UPDATE(G, \alpha)$ is sound, complete, and terminating. □

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

Theorem 2. *Under ABox deletions, $UPDATE(G, \alpha)$ is sound, complete, and terminating.*

Proof Sketch

As shown in [16,15,10], the tracing function captures all *change events* in G that were caused in part by α. It can be shown that by undoing all events 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. This holds because the tracing algorithm is shown to be complete [16,15,10], thus *all* necessary components will be rolled-back. Because $UPDATE(G, \alpha)$ performs this rollback, it can be shown that the updated completion graph is a possible intermediate state of the a completion graph for the KB after the deletion. While this graph is potentially incomplete (e.g., due to blocking), reapplying expansion rules guarantees (by [12,13]) the algorithm will arrive at some completion graph that could be obtained by simply applying the completion rules to $\mathsf{K} \cup \{\alpha\}$. It can therefore be shown that under ABox deletions, $UPDATE(G, \alpha)$ is sound, complete, and terminating. □

5 Implementation and Evaluation

We have implemented the approach presented in this paper in an open source OWL-DL reasoner, Pellet [22]. In order to evaluate the algorithm, we have performed an emperical evaluation using two different KBs with large ABoxes - the Lehigh University Benchmark (LUBM)[2] and AKT Reference Ontology [3].

For the LUBM test case, three experiments were run over three different KBs consisting of one, two, and lastly three universities created by 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 dataset. Update sizes include single axiom, twenty-five axioms, and fifty axioms (individuals and/or roles). Each test was averaged over twenty-five times iterations. Expressivity and KB statistics are provided in Table 1.

Results for additions and deletions in the LUBM test are presented in Figures 2 and 3 (timing results are shown in milliseconds and the scale is logarithmic). We note

[2] LUBM Ontology: http://swat.cse.lehigh.edu/projects/lubm/
[3] AKT Ontology: http://www.aktors.org/publications/ontology/

Table 1. LUBM and AKT Portal Dataset Statistics

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

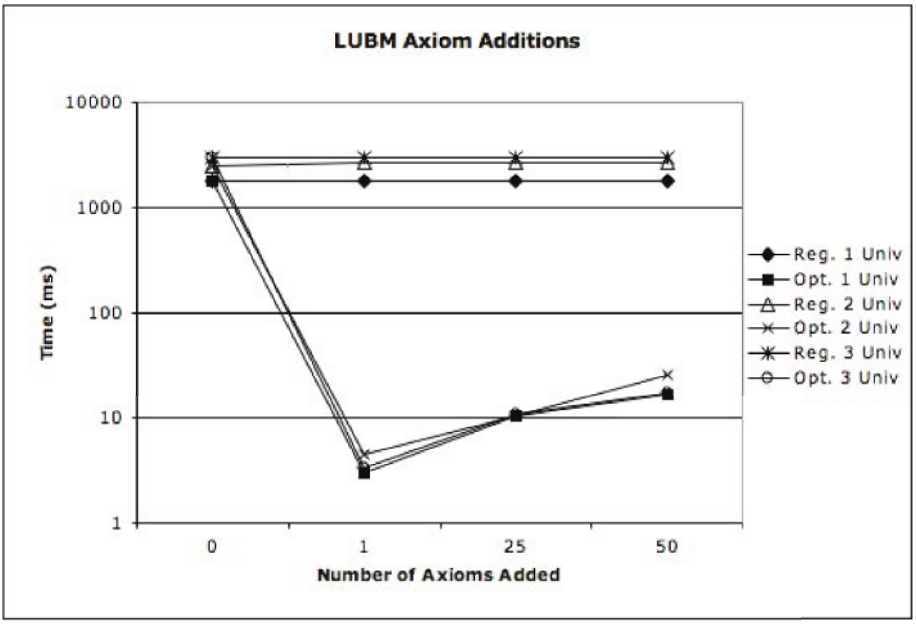

Fig. 2. Addition Updates of LUBM datasets

that the '0' axiom value represents the initial consistency check. In both versions of the reasoner, initial consistency checks are comparable. However for both update types (additions and deletions), 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 fired. It can also be observed that as the update size is increased, the performance of the update approach scales well. This provides direct empirical evidence for the effectiveness of the update algorithm.

In a second evaluation, two datasets[4] 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

[4] Hyphen-REA: http://www.hyphen.info/rdf/hero_complete.zip

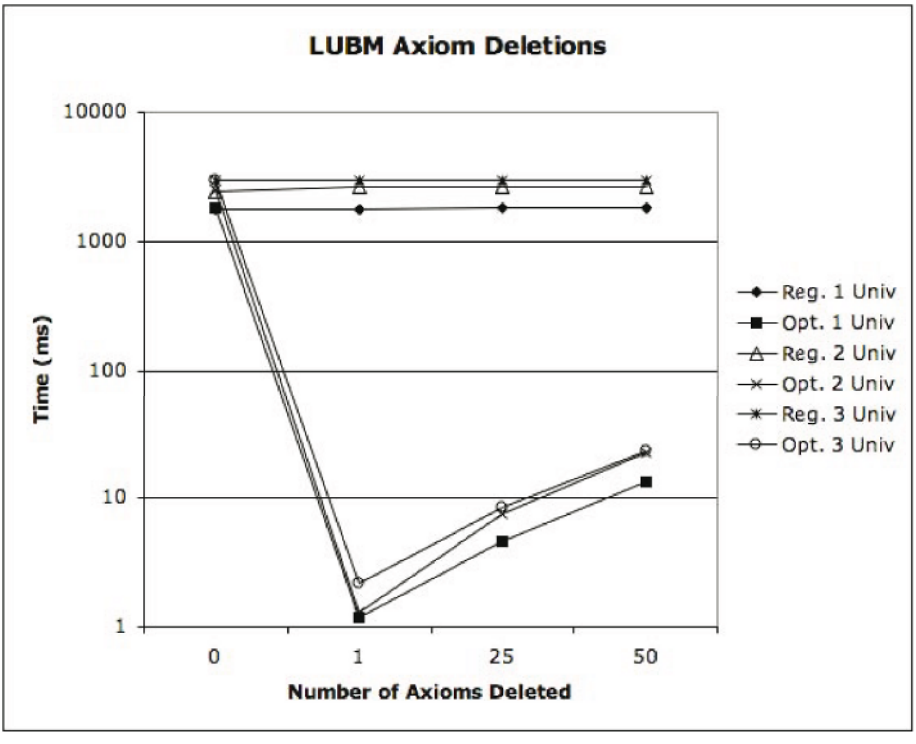

Fig. 3. Deletion Updates of LUBM datasets

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 (as shown in Figures 4 and 5). It is interesting to observe that the performance of the deletion updates is slightly better in the LUBM test cases for larger sized updates. 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 approach.

6 Discussion and Future Work

One limitation of the approach presented in the work is related to potential overhead introduced by the algorithm, specifically related to tracing. However, we point out here that in [16] the tracing approach was shown to introduce small memory overhead and only marginally increase the running time of the normal reasoning procedure. For example, in the Tambis [5] ontology, tracing introduced only 56ms to the running time and 3.65mb memory overheard [16]. Therefore, we feel the approach is acceptable due to the observed performance improvements.

[5] Tambis ontology: http://www.cs.man.ac.uk/ horrocks/OWL/Ontologies/tambis-full.owl

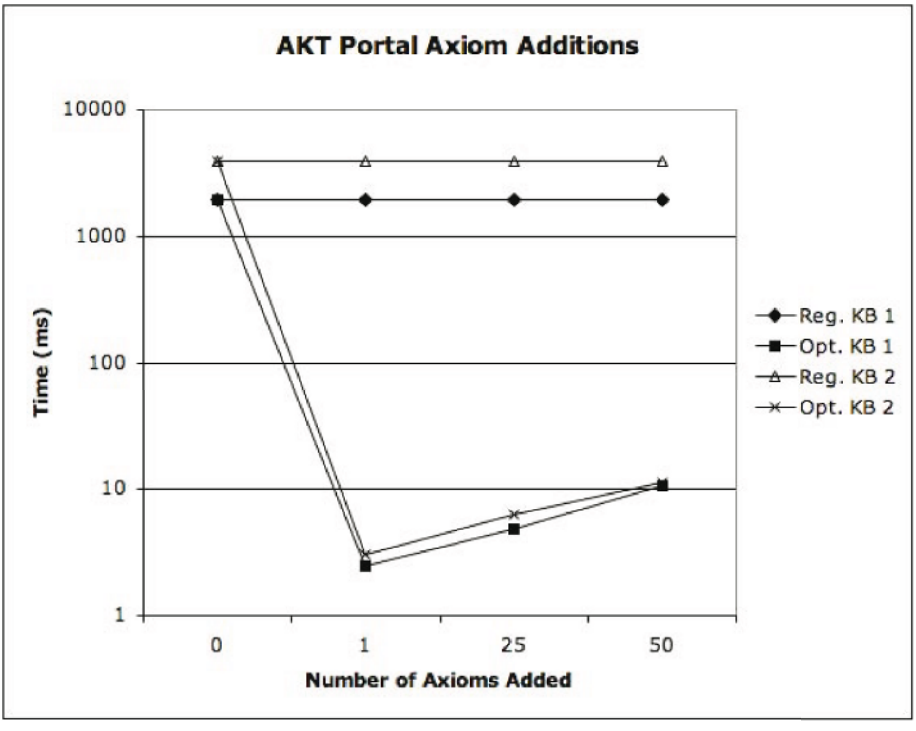

Fig. 4. Addition Updates of AKT datasets

The approach presented in this work is applicable to the Description Logics $\mathcal{SHOQ}$ and $\mathcal{SHIQ}$; this is primarily due to fact that there is no expansion rule ordering imposed by the tableau algorithms [12,13]. We are currently working to extend the approach to $\mathcal{SHOIQ}$ (and therefore all of OWL-DL), which will be addressed in future work.

While we achieved dramatic results for consistency checking under syntactic ABox updates, one may wish to update TBox axioms as well. This presents the additional issue of rolling back through pre-processing steps, such as absorption. We are currently investigating this problem and plan to address it in future work.

In the current approach, if the KB is inconsistent after the update, nothing is done to resolve the inconsistency. As future work, we are working towards developing a revision algorithm for OWL-DL KBs; with such a technique, inconsistencies resulting after the update would be resolved using a revision operator.

This work only directly addresses consistency checking under ABox changes; however, standard reasoning tasks in DL reasoners, including classification, realization, and query answering are all reducible to KB consistency checking [3]. Currently, we are investigating the utility of the update algorithm for generalized reasoning services. We note here that initial results on leveraging the approach presented in this paper for continuous conjunctive query answering demonstrates orders of magnitude improvements in performance.

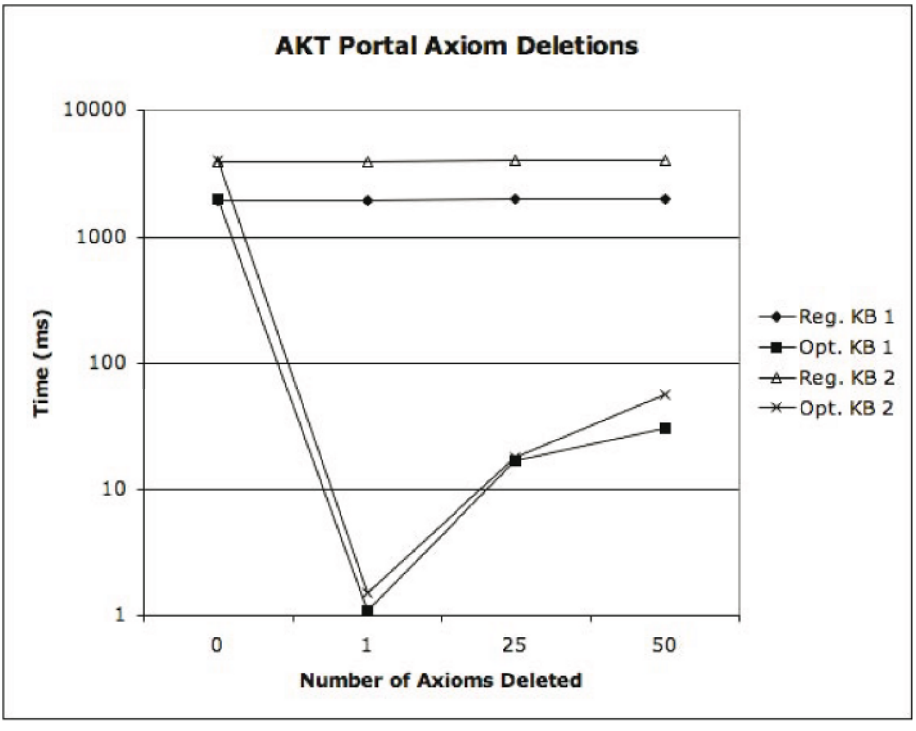

Fig. 5. Deletion Updates of AKT datasets

7 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 paralleled to view maintenance [7] and truth maintenance [4]; however we deal with a more expressive logic and a different proof theory.

In this work, we have leveraged and extended previous work in axiom tracing [2,15,16]. [2] introduces the notion of axiom pinpointing for the purpose of computing extensions of default theories. [15,16] extends [2] to support a more expressive logic. As discussed earlier, we have further extended [15,16] as further effects of axioms were needed to be identified.

Recently, there has been interest in specifying formal update semantics for descriptions logics [19,23]. Additionally, [5,6] has investigated applying traditional AGM belief revision [1] theory to DL knowledge bases; the authors show however, that in certain expressive DLs (including those considered in this work) the AGM postulates for contraction cannot be satisfied. This finding does not impact this work, as we assume syntactic updates of asserted axioms. Further, this work is independent of belief revision and formal update semantics as we are concerned with maintaining the internal state of the reasoner.

8 Conclusion

Current Description Logic reasoners are traditionally oriented toward providing reasoning services for relatively static ontologies. However, there are numerous use cases in which the ontology itself is in flux, requiring frequent updates. This includes prominent Web services frameworks (e.g., OWL-S), in which Web ontologies are used for service discovery and matchmaking. In such settings devices register or deregister their descriptions quite rapidly. Additionally, Semantic Web portals often allow content authors to modify or extend the ontologies which organize their site structure or page content. Lastly, there has been recent interest in utilizing DL reasoning for the purpose of matching published content with subscription requests [8,9,26,17]. When disseminating time sensitive information efficient reasoning for frequently changing KBs will be critical in achieving practical DL-based approaches.

While there exists such use cases for reasoning under changing data, current DL reasoning algorithms have been developed considering relatively static knowledge bases. In this paper, we have presented an algorithm for updating tableau completion graphs for the Description Logics $\mathcal{SHIQ}(\mathcal{D})$ and $\mathcal{SHOQ}(\mathcal{D})$ 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, in which our initial results are very promising as they demonstrate orders of magnitude performance improvement.

Acknowledgments

This work was supported in part by grants from Fujitsu, Lockheed Martin, NTT Corp., Kevric Corp., SAIC, the National Science Foundation, the National Geospatial Intelligence Agency, DARPA, US Army Research Laboratory, and NIST. We would also like to thank Aditya Kalyanpur, Yarden Katz, and Vladimir Kolovski for all of their contributions to this work.

References

1. Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, 1985.
2. F. Baader and B. Hollunder. Embedding defaults into terminological representation systems. *J. Automated Reasoning*, 14:149–180, 1995.
3. 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.
4. Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 1979.
5. G. Flouris, D. Plexousakis, and G. Antoniou. On applying the agm theory to dls and owl. In *4th International Semantic Web Conference (ISWC 2005)*, 2005.

6. G. Flouris, D. Plexousakis, and G. Antoniou. Updating description logic using the agm theory. In *7th International Symposium on Logical Formalizations of Commonsense Reasoning*, 2005.
7. A. Gupta and I. Mumick. Materialized views: Techniques, implementation, and applications. In *MIT press*, 1999.
8. V. Haarslev and R. Moller. Description logic systems with concrete domains: Applications for the semantic web. In *In Int. Workshop on KR meets Databases, 2003.*, 2003.
9. V. Haarslev and R. Möller. Incremental query answering for implementing document retrieval services. In *Proceedings of the International Workshop on Description Logics (DL-2003), Rome, Italy, September 5-7*, pages 85–94, 2003.
10. Christian Halaschek-Wiener, Aditya Kalyanpur, and Bijan Parsia. Extending tableau tracing for abox updates. In *UMIACS Tech Report*, 2006. http://www.mindswap.org/papers/2006/aboxTracingTR2006.pdf.
11. I. Horrocks. Implementation and optimisation techniques. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 313–355. Cambridge University Press, 2003.
12. 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.
13. 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.
14. 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.
15. Aditya Kalyanpur. Debugging and repair of owl ontologies. In *Ph.D. Dissertation, University of Maryland, College Park*. http://www.mindswap.org/papers/2006/AdityaThesis-DebuggingOWL.pdf.
16. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging unsatisfiable classes in owl ontologies. In *Journal of Web Semantics - Special Issue of the Semantic Web Track of WWW2005*, 2005.
17. L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology, 2003.
18. T. Liebig and O. Noppens. Ontotrack: Combining browsing and editing with reasoning and explaining for owl lite ontologies. In *Proceedings of the 3rd International Semantic Web Conference (ISWC) 2004*, Japan, November 2004.
19. 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.
20. Bernhard Nebel. Base revision operations and schemes: Semantics, representation, and complexity, 1994.
21. Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic matching of web services capabilities. In *The First International Semantic Web Conference*, 2002.
22. Bijan Parsia and Evren Sirin. Pellet: An owl dl reasoner. In *Third International Semantic Web Conference - Poster*, 2004.
23. Mathieu Roger, Ana Simonet, and Michel Simonet. Toward updates in description logics. In *International Workshop on Knowledge Representation meets Databases*, 2002.
24. Evren Sirin, Bijan Parsia, and James Hendler. Composition-driven filtering and selection of semantic web services. *IEEE Intelligent Systems*, 19(4):42–49, 2004.

25. Katia Sycara, Massimo Paolucci, Anupriya Ankolekar, and Naveen Srinivasan. Automated discovery, interaction and composition of semantic web services. *Journal of Web Semantics*, 1(1):27–46, 2003.
26. Michael Uschold, Peter Clark, Fred Dickey, Casey Fung, Sonia Smith, Stephen Uczekaj Michael Wilke, Sean Bechhofer, and Ian Horrocks. A semantic infosphere. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 882–896. Springer, 2003.