A Combined Approach to Incremental Reasoning for EL Ontologies

Yuan Ren, Jeff Z. Pan(✉), Isa Guclu, and Martin Kollingbaum

Department of Computing Science, University of Aberdeen, Aberdeen, UK
jeff.z.pan@abdn.ac.uk

Abstract. Due to the dynamic nature of knowledge and data in semantic applications, ontology incremental reasoning technologies are essential for ontology management systems. Nowadays, many proposed incremental reasoning solutions and implemented systems apply forward chaining completion algorithms to handle the removal and addition of axioms. In this paper, we propose a novel approach to ontology incremental reasoning that combines forward and backward chaining completion for $\mathcal{EL}$. Compared to existing work, this approach can be applied with or without bookkeeping, does not affect parallelisation or tractability, and reduces the effort for re-deriving the over-deleted results both theoretically and empirically.

1 Introduction

Ontologies are widely used in many different application domains to support knowledge management. In order to facilitate automatic processing of ontologies, today's *de facto* standard ontology languages, the Web Ontology Languages (OWLs), are based on a family of Description Logics [1] (DLs). There are some profiles for OWL 2, including EL, QL and RL. Using DLs, ontologies can be regarded as a set of logical axioms.

Ontologies and their corresponding reasoning results are usually considered static. However, with the expanding applications of ontologies, such a paradigm has been challenged [13,16]. In many scenarios, ontologies are subject to rapid changes [12].

The dynamics of ontologies have brought many new research challenges, such as the design of new knowledge representation and query languages [3,5], the development of new reasoning services [10,12] and the development of stream benchmarks [18]. In this paper, we are particularly interested in the development of *incremental reasoning* technologies that update reasoning results affected by the updating of the ontology without naively re-computing all results. In order to reuse previously computed results, many existing approaches [4,8,11,17,21] adopted the Delete and Re-derive (DRed) strategy [7]. With DRed, an incremental reasoner first over-estimates and over-deletes the results affected by the deleted original axioms, unaffected results are preserved. It then re-derives the over-deleted results that can be entailed by the preserved axioms. The authors of [20] pointed out that DRed needs to examine all preserved results during

M. Ortiz and S. Schlobach (Eds.): RR 2016, LNCS 9898, pp. 167–183, 2016.
DOI: 10.1007/978-3-319-45276-0_13

re-derivation and proposed to completely avoid re-derivation by using counting to pinpoint the affected results. Despite of the different mechanisms, all these approaches adopt forward chaining consequence-based algorithms to compute results. For Datalog-based systems, Motik et al. [14] proposed a Backward/Forward (B/F) algorithm for reducing the work done by combining backward and forward chaining to efficiently update the materialization incrementally. In their approach, B/F continuously outperformed the DRed algorithm up to a threshold of 12 % updates in the initial ontology

In this paper, we present a novel DRed approach to ontology incremental reasoning by combining forward chaining and backward chaining of consequence-based algorithms. It has several advantages: (1) It can be applied with either bookkeeping or non-bookkeeping. (2) It helps to reduce the volume of over-deletion and re-derivation in the DRed strategy, as we will show both theoretically and empirically. (3) It can be parallelised and allows the use of multiple computational cores with shared main memory, when applied in parallel reasoners. (4) It works for OWL 2 EL and can be applied to any knowledge representation that supports a consequence-based procedure. When applied with a tractable algorithm, our approach is also tractable. In this paper, we will focus on (1) and (2) with OWL 2 EL [2].

2 Background

In this section, we introduce the most relevant notions of syntax and semantics of DLs. See [1] for a more thorough introduction on DLs.

Briefly, an ontology $\mathcal{O}$ is a set of DL axioms, containing a TBox (schema part of $\mathcal{O}$) and ABox (data part of $\mathcal{O}$). An axiom α is *entailed* by an ontology $\mathcal{O}$, written $\mathcal{O} \models \alpha$, *iff* all models of $\mathcal{O}$ satisfy α. $J_{\mathcal{O}}(\alpha) \subseteq \mathcal{O}$ is a *(minimal) justification* of α *iff* $J_{\mathcal{O}}(\alpha) \models \alpha$ and $J' \not\models \alpha$ for all $J' \subset J_{\mathcal{O}}(\alpha)$. The algorithms presented in this paper handle both TBox and ABox axioms.

A consequence-based algorithm usually consists of two closely related components: a set of completion rules and a serialised forward-chaining procedure to apply the rules. For example, below is a completion rule for the DL $\mathcal{EL}^+$, in which $\sqsubseteq^*$ is the transitive, reflexive closure of $\sqsubseteq$ in $\mathcal{O}$.

$$\mathbf{R}_{\exists}\frac{E \sqsubseteq \exists R.C, C \sqsubseteq D, R \sqsubseteq^* S}{E \sqsubseteq \exists S.D} : \exists S.D \text{ occurs in } \mathcal{O}$$

We say that some axioms in the ontology can be used as premises (consequences) of a rule when they satisfy the syntactic form specified by the premises (consequences) of the rule. When it is clear from context, we also simply call these axioms premises (consequences) of the rule.

Given an ontology, a consequence-based algorithm repeatedly applies all completion rules in the rule set until no more rule can be applied to compute the *completion closure*:

Definition 1 ***(Completion Closure).*** *For a set of axioms S and a completion rule set R, the immediate results of applying R on S, denoted by $R(S)$ or $R^1(S)$,*

is the set of axioms that are either in S, or can be derived as consequence from premises in S by a single rule in R.

Let $R^{n+1}(S) = R(R^n(S))$ for $n \geq 1$, then the completion closure *of S w.r.t. R, denoted by $R^*(\mathcal{O})$, is some $R^n(S)$ s.t. $R^n(S) = R(R^n(S))$.*

A rule set R converges *if for any $\mathcal{O}$, $R^*(\mathcal{O})$ exists.*

It can be show that the following properties hold:

Lemma 1. *Let $S_{(i)}$ be sets of axioms, R a set of completion rules and $n \geq 1$:*

$$S_1 \subseteq S_2 \rightarrow R^n(S_1) \subseteq R^n(S_2) \tag{1}$$

$$R^*(R^n(S)) = R^*(S) \tag{2}$$

$$R^*(S_1 \cup S_2) = R^*(R^n(S_1) \cup S_2) \tag{3}$$

The step of deriving consequences from premises using a rule is an *execution* of the rule. The computation of closure can then be described with the help of a list L in the following algorithm *FCC (Forward Chaining Completion)*.

Forward Chaining Completion:
$FCC(L, S, R)$
INPUT: a list of axioms to be processed L, a set of processed axioms S, a completion rule set R
OUTPUT: a set of processed axioms S

```
1: while L ≠ ∅ do
2:    get an element α ∈ L
3:    L := L \ {α}, S := S ∪ {α}
4:    for each rule in R do
5:       if α can be used as a premise, and all other premises α_2, ..., α_n are in
         S, and consequence β is not in S ∪ L then
6:          execute rule and add β into L
7: return S
```

For each $\alpha \in L$, FCC checks if it can be used to execute a rule with other axioms in S to infer $\beta \notin S \cup L$, which implies that β has not been processed or derived yet. If that is the case, the rule will be executed and β will be added into L. In any case, α will be moved from L to S.

It can be shown that $R^*(L) = FCC(L, \emptyset, R)$:

Lemma 2. *If $R(S) \subseteq S \cup L$, then $R^*(S \cup L) = FCC(L, S, R)$.*

Since $R^*(\emptyset) \subseteq L \cup \emptyset$, we have $R^*(L) \subseteq FCC(L, \emptyset, R)$. With the above procedure, consequence-based algorithms can be used to perform ontology reasoning such as classification and materialisation.

In this paper, we focus on incremental reasoning. We consider an ontology sequence $(\mathcal{O}_1, t_1), \ldots, (\mathcal{O}_n, t_n)$, in which $\mathcal{O}_i$ are DL ontologies and $t_1 < \cdots < t_n$ are time points. The change from $\mathcal{O}_i$ to $\mathcal{O}_{i+1}$ is an update of the ontology. In this

paper, given an $\mathcal{O}_i$ and its following snapshot $\mathcal{O}_{i+1}$, we address the problem of computing the updated completion closure $FCC(\mathcal{O}_{i+1}, \emptyset, R)$. There are two different approaches to solve this problem. One is *Naive Reasoning*, which recomputes all results completely. The other is *Incremental Reasoning*, which attempts to re-use the results of $FCC(\mathcal{O}_i, \emptyset, R)$ to compute $FCC(\mathcal{O}_{i+1}, \emptyset, R)$, without completely re-computing $FCC(\mathcal{O}_i, \emptyset, R)$. The later is the focus of this paper.

3 Technical Motivation

To deal with incremental reasoning, one key challenge is to handle the deletion of original axioms. The authors of [21] first adopted the Delete and Rederive (DRed) strategy [7,19] from traditional data stream management systems and applied it on ontology incremental reasoning. A DRed approach first *over-deletes* all the potential consequences of the original deletion. Other results will be preserved. It then *re-derives* the over-deleted consequences that can be derived by the preserved results. It finally performs reasoning to deal with the new facts, which can be realised with the same mechanism we just introduced. Such a mechanism has also been adopted by all the existing incremental reasoning approaches. Hence, in this paper we will focus on the optimisation of the over-deletion and re-derivation.

Let $\mathcal{O}$ be an ontology, $R^*(\mathcal{O}) = FCC(\mathcal{O}, \emptyset, R)$ be its completion closure w.r.t. R, $Del \subseteq \mathcal{O}$ be a set of axioms to remove. A DRed approach first identifies a set of *valid over-deletion* $OD \subseteq R^*(\mathcal{O})$ w.r.t. Del:

Definition 2 ***(Valid Over-deletion).*** *Let $\mathcal{O}$ be an ontology, $Del \subseteq \mathcal{O}$ a set of axioms to remove, R a completion rule set, an over-deletion OD of $R^*(\mathcal{O})$ w.r.t. Del is valid if:*

1. *$\forall \alpha \in R^*(\mathcal{O})$, if for every $\mathcal{J}_{\mathcal{O}}(\alpha)$ it is true that $\mathcal{J}_{\mathcal{O}}(\alpha) \cap Del \neq \emptyset$, then $\alpha \in OD$.*
2. *$\forall \alpha \in R^*(\mathcal{O} \setminus Del)$, there is some $\mathcal{J}_{\mathcal{O}}(\alpha)$ such that $\mathcal{J}_{\mathcal{O}}(\alpha) \cap OD = \emptyset$.*

The first condition ensures that OD over-deletes all entailments that can only be inferred from some axioms in Del. The second condition ensures that any entailment of $O \setminus Del$ is also entailed by $R^*(\mathcal{O}) \setminus OD$. A DRed approach then re-derives any axiom $\alpha \in OD$ if there is some $\mathcal{J}_{\mathcal{O}}(\alpha)$ s.t. $\mathcal{J}_{\mathcal{O}}(\alpha) \cap Del = \emptyset$. Different DRed or non-DRed incremental reasoning approaches differ primarily on how they identify the over-deleted results and how they perform re-derivation. We introduce them w.r.t. their re-derivation mechanism:

– *Global Re-derivation:* There are a few variants, of the global re-derivation approach, including [4,11,17,21]. These global re-derivation DRed approaches have two major limitations: (1) Some of these approaches require bookkeeping, e.g., a TMS [17], to identify the valid over-deletion. Such bookkeepings will impose performance and resource-consumption overhead; (2) The re-derivation has to go through all remaining axioms $R^*(\mathcal{O}) \setminus OD$ to ensure the completeness of results, even if many of them cannot infer further entailments. A TMS is a loopless directed graph in which nodes denote the axioms in

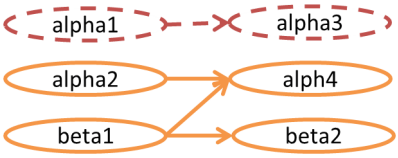

Fig. 1. Over-deletion with a TMS

the completion closure, and edges connect premise and side condition axioms to consequence axioms. When some original axioms are deleted, all axioms to which the deleted axioms have paths in the TMS will be over-deleted. Consider the example:

Example 1. Figure 1 shows a TMS in over-deletion. In this figure, $\mathcal{O} = \{\alpha_1, \alpha_2, \beta_1\}$ and $R^*(\mathcal{O}) = \mathcal{O} \cup \{\alpha_3, \alpha_4, \beta_2\}$.

When $Del = \{\alpha_1\}$ is deleted, $OD = Del \cup \{\alpha_3\}$ since according to the TMS, α_3 is the only entailment connected from α_1 in the TMS.

– *Local Re-derivation:* A recent work [8] proposed a non-bookkeeping DRed approach to address the above limitations. The key point is to exploit the independent nature of different contexts to facilitate the over-deletion and re-derivation. Nevertheless, this approach also has limitations: (1) It relies on the context in rules so it is not applicable to consequence-based algorithms without context; (2) It almost always over-deletes more axioms than necessary. This approach first re-runs a similar forward chaining procedure as in algorithm *FCC* to identify a set of entailments $DEL = \{C \sqsubseteq D | C \sqsubseteq D \in R^*(\mathcal{O}) \setminus (\mathcal{O} \setminus Del)$ and can be directly or indirectly derived from premises in $Del\}$. The computation of DEL is similar to the over-deletion proposed in [11]. It then computes $Broken = \{C \sqsubseteq E | C \sqsubseteq E \in R^*(\mathcal{O}), C \sqsubseteq D \in DEL\}$, i.e. all derived GCIs who share a LHS (left hand side) context with some GCI that can be derived from the removed axioms. Below is an example:

Example 2. Figure 2 shows a closure similar as the one in Fig. 1. Now the closure is partitioned into two contexts. α_is all have context C_1 and β_is all have context C_2. Suppose we still have $Del = \{\alpha_1\}$ and $DEL = Del \cup \{\alpha_3\}$, since α_2 and α_4 also belongs to the same context, we have $\{\alpha_2, \alpha_4\} \subseteq Broken$.

– *No Re-derivation:* In order to completely avoid the re-derivation in DRed, the authors of [20] apply the counting strategy proposed in [7] to pinpoint the axioms that have to be removed from $R^*(\mathcal{O})$. This approach is conceptually and empirically more efficient than DRed when dealing with removal of axioms.

Example 3. The upper part of Fig. 3 shows a TMS similar as the one in Fig. 1. The main difference is that now the TMS recognised that α_3 can not only be derived from α_1, but also α_2. Hence its count $N(\alpha_3) = 2$.

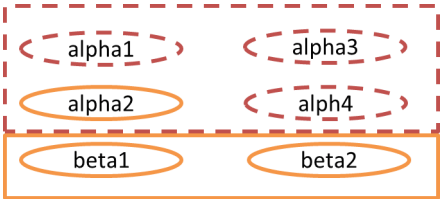

Fig. 2. Over-deletion with context

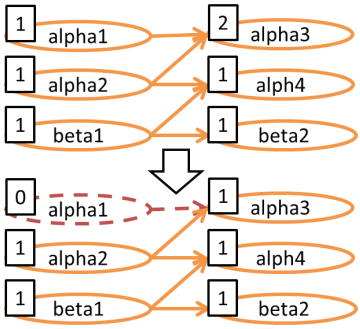

Fig. 3. Over-deletion with counting

The lower part of Fig. 3 shows how the deletion works. When $Del = \{\alpha_1\}$, $N(\alpha_1) = 0$. Consequently $N(\alpha_3) = 1$. Since $N(\alpha_3) \neq 0$, α_3 will not be deleted and its derivation from α_2 is still preserved.

However, it needs to make a trade-off between efficiency and quality of results: (1) If the independent rule executions for each entailment are not thoroughly and precisely identified, then this approach might not yield exactly the same results as naive reasoning. (2) In order to obtain and maintain all possible rule executions for each entailment, this approach essentially computes all justifications for all entailments in the closure. This is known to be expensive even for $\mathcal{EL}$.

In this paper, we do not want to increase the complexity of incremental reasoning in comparison to naive reasoning. We want the results to be exactly the same as naive reasoning. Therefore, we will use DRed instead of the counting strategy. We also want our approach to not to rely on contexts, but be parallelisable with contexts, so that it is applicable with both bookkeeping and non-bookkeeping methods.

4 Combining Forward and Backward Chaining

In order to avoid unnecessary axiom over-deletion and re-derivation, it is necessary to develop a re-derivation mechanism that focuses only on the preserved entailments that can be used to infer over-deleted axioms.

Full Backward Chaining Re-derivation

One way to achieve our goal is a full backward chaining procedure: A backward chaining procedure starts from the entailments that are attempted to be re-derived. It then checks which potential premises in the original closure can be used to derive such an entailment. If all the premises have been preserved during over-deletion or re-derived during re-derivation, then the target axiom can be re-derived. Otherwise, the algorithm can try to re-derive the potential premises recursively. Eventually, this procedure can re-derive all entailments that can be inferred from the preserved axioms. Such a procedure can be described with the following algorithms:

Full Backward Chaining Re-derivation:
$fBCRD(L, S, R)$
INPUT: a list of axioms to be re-derived L, a partial closure S, a completion rule set R
OUTPUT: a set of re-derived axioms *Rederived*

1: $Rederived := \emptyset$
2: **while** $L \neq \emptyset$ **do**
3: get $\alpha \in L$
4: $L := L \setminus \{\alpha\}$
5: $fTest(L, S, Rederived, \{\alpha\}, \alpha, R)$
6: **return** *Rederived*

Given a closure after over-deletion S, a list of over-deleted axioms L and a rule set R, Algorithm $fBCRD(L, S, R)$ finds out all axioms in L that can be directly or indirectly re-derived from S, i.e. $fBCRD(L, S, R) = L \cap R^*(S)$:

1. In Step-1 it first initialises the set of re-derived entailments, which is empty initially.
2. From Step-2 to Step-5, it iteratively tests each entailment $\alpha \in L$ until L is empty. Such a testing is performed by a sub-procedure Algorithm *fTest*.

Full Test:
$fTest(L, S, Rederived, Testing, \beta, R)$
INPUT: a list of axioms to be re-derived L, a partial closure S, a set of re-derived axioms *Rederived*, a set of axioms being tested $Testing$, an axiom to be tested β and a set of completion rules R
OUTPUT: nothing, but L and *Rederived* will be altered during the execution of the algorithm

1: **if** $\beta \notin Rederived$ **then**
2: **for** each $rule \in R$ **do**
3: **if** β can be used as the consequence of *rule*, and all premises $A = \{\alpha_1, \ldots, \alpha_n\}$ of *rule* are in $S \cup L \cup Rederived \setminus Testing$ **then**

4: **while** $A \cap L \neq \emptyset$ **do**
5: get $\alpha \in A \cap L$
6: $fTest(L,S,Rederived,Testing \cup \{\beta\},\alpha,R)$
7: **if** $A \subseteq S \cup Rederived$ **then**
8: $Rederived := Rederived \cup \{\beta\}$
9: $L := L \setminus \{\beta\}$
10: **return**

Given S, L, $Rederived$, R and a set of axioms being tested $Testing$, Algorithm *fTest* will check if an entailment β can be re-derived with R from entailments in S. If β can be re-derived, the algorithm will extend $Rederived$ accordingly. To test the possibility of re-derivation, the algorithm will recursively test the premises of β in L.

Let $R^*(\mathcal{O})$ be a completion closure of ontology $\mathcal{O}$ w.r.t. rule set R and OD be the set of over-deleted axioms, with the above algorithms, re-derivation can be performed with $fBCRD(OD, R^*(\mathcal{O}) \setminus OD, R)$. It can be shown that the above procedure produces the correct and complete re-derivation results:

Lemma 3. *Let $\mathcal{O}$ be an ontology, $S = R^*(\mathcal{O}) = FCC(\mathcal{O}, \emptyset, R)$ be the completion closure of $\mathcal{O}$ w.r.t a set of completion rules R, $Del \subseteq \mathcal{O}$ be a set of deleted axioms, $OD \subseteq S$ be a valid over-deletion w.r.t. Del, then:*

$$(S \setminus OD) \cup fBCRD(OD, S \setminus OD, R) = R^*(\mathcal{O} \setminus Del).$$

Combined Forward and Backward Chaining Re-derivation. The full backward chaining approach introduced in the previous subsection can be further optimised. Particularly, in the presented procedure, the testing of the same axiom may be invoked multiple times.

In this section, we present a more efficient variant of the previous procedure that eliminates the redundant testings. The key-point is to combine forward and backward chaining in re-derivation:

1. Assuming we have a completion closure $S = R^*(\mathcal{O})$ and a set of over-deleted axioms OD, the purpose of re-derivation is to compute $R^*(S \setminus OD)$.
2. Forward chaining re-derivation achieves this by computing $FCC(S \setminus OD, \emptyset, R)$ either globally or locally, and may process unnecessary entailments.
3. Instead, we only need to find $L' = R(S \setminus OD) \setminus (S \setminus OD)$, and then compute $FCC(L', S \setminus OD, R)$. Since we have $R(S \setminus OD) \subseteq (S \setminus OD) \cup L'$, according to Lemma 2, results of $FCC(L', S \setminus OD, R)$ is the same as $R^*((S \setminus OD) \cup L')$. Since $L' = R(S \setminus OD) \setminus (S \setminus OD)$, we have $(S \setminus OD) \cup L' = R(S \setminus OD)$. According to Property (2) of Lemma 1, $R^*((S \setminus OD) \cup L') = R^*(R(S \setminus OD)) = R^*(S \setminus OD)$.

The above procedure is the forward chaining part. The $L' = R(S \setminus OD) \setminus (S \setminus OD)$ will be computed by backward chaining. It can be achieved with a procedure similar to $fBCRD$:

Backward Chaining Re-derivation:
$BCRD(L, S, R)$
INPUT: a list of axioms to be re-derived L, a partial closure S, a completion rule set R
OUTPUT: a set of re-derived axioms *Rederived*

1: $Rederived := \emptyset$
2: **for** each $\alpha \in L$ **do**
3: $\quad Test(S, Rederived, \alpha, R)$
4: **return** *Rederived*

Test:
$Test(S, Rederived, \beta, R)$
INPUT: a partial closure S, a set of re-derived axioms *Rederived*, an axiom to be tested β and a set of completion rules R
OUTPUT: nothing, but *Rederived* will be altered during the execution of the algorithm

1: **for** each *rule* $\in R$ **do**
2: $\quad$ **if** β can be used as the consequence of *rule*, and all premises $A = \{\alpha_1, \ldots, \alpha_n\}$ of *rule* are in S **then**
3: $\quad\quad Rederived := Rederived \cup \{\beta\}$
4: $\quad\quad$ **return**

As we can see, the procedure is different from the previous full backward chaining re-derivation on the following aspects:

1. Instead of testing axioms with algorithm *fTest*, a new algorithm *Test* is used.
2. *Test* no longer recursively checks if a premise is re-derivable when it is not immediately available in S. Instead, it only checks if all premises are in S, which is the preserved partial closure. Hence the set $Testing$ is not needed, because a tested axiom will not be used as premise to re-derive another tested axiom.
3. As a consequence, $BCRD(L, S, R)$ will compute $L \cap R(S)$, namely all axioms in L that can be directly re-derived from S.

Therefore, for a completion closure S and a valid over-deletion OD, we have $R(S \setminus OD) \setminus (S \setminus OD) = BCRD(OD, S \setminus OD, R)$. Combining with the forward chaining part mentioned above, re-derivation of $R^*(S \setminus OD)$ can be achieved:

Theorem 1. *Let $\mathcal{O}$ be an ontology, $S = R^*(\mathcal{O}) = FCC(\mathcal{O}, \emptyset, R)$ be the completion closure of $\mathcal{O}$ w.r.t. a set of completion rules R, $Del \subseteq \mathcal{O}$ be a set of deleted axioms, $OD \subseteq S$ be a valid over-deletion w.r.t. Del, then:*

$$R^*(\mathcal{O} \setminus Del) = FCC(BCRD(OD, S \setminus OD, R), S \setminus OD, R).$$

Proof (Sketch)**.** We can first show that $R^*(\mathcal{O} \setminus Del) = R^*(S \setminus OD)$. Hence we only need to prove $R^*(S \setminus OD) = FCC(BCRD(OD, S \setminus OD, R), S \setminus OD, R)$. According to Lemma 2, we only need to prove that $R(S \setminus OD) \subseteq (S \setminus OD) \cup BCRD(OD, S \setminus OD, R)$. For any $\alpha \in R(S \setminus OD)$, it is either trivially in $(S \setminus OD$, or it will be added into *Rederived* in Step-2 of Algorithm *Test*. □

When completion rule set R is tractable, this procedure is also tractable since both $BCRD$ and FCC will be tractable. This suggests that our re-derivation itself does not affect the tractability of reasoning in general. When the completion rule set is intractable, our approach is as complex as forward chaining completion. Although not affecting the worst case computational complexity, conceptually, such a combined forward and backward chaining re-derivation has a minimal problem space (the over-deleted entailments) and a small search space (all premises must be preserved). The re-derivation will only examine the over-deleted axioms *once* in the backward chaining stage and only process the re-derived axioms *once* in the forward chaining stage. These characteristics make the combined re-derivation more efficient than the full forward chaining or full backward chaining re-derivation, especially when the over-deleted entailments are much less than the preserved entailments.

Our approach does not rely on bookkeeping dependencies between premise and consequence axioms. When performing Step-2 of Algorithm *Test*, an implemented system only needs to identify one candidate premise α, and then it can use α in the same way as in Step-5 of Algorithm FCC to find other premises. The identification of α can be realised by exploiting the structural relationships between premise and consequence of each rule. For example, in order to re-derive α (e.g., $E \sqsubseteq \exists S.D$) with backward chaining of rule $\mathbf{R}_{\exists}$, the reasoner only needs to search for a preserved entailment β (e.g., $E \sqsubseteq \exists R.C$) with the same LHS as α s.t. another γ (e.g., $C \sqsubseteq D$) whose LHS is the RHS filler of β, and whose RHS is the same as α is preserved, and $R \sqsubseteq^* S$ holds. In general, if FCC can be performed without bookkeeping, our approach can be performed without bookkeeping. Nevertheless, our approach can also be augmented with bookkeeping in the same way as FCC. Our approach also does not rely on context in rules. Our approach can also be modified to calculate the counts of entailments, but this is clearly out of the scope of this paper. We will leave it to our future work.

5 Experimental Evaluation

In order to evaluate the usefulness and performance of our approach, we conducted an empirical evaluation to find out:

1. Whether our approach can be used to reduce the number of axioms that are processed, in comparison to global re-derivation and local re-derivation.
2. Whether our approach can be used to achieve efficient incremental reasoning in terms of execution time and memory consumption.

Implementation

For evaluation purposes, we implemented the following approaches:

1. In order to compare with the *naive reasoning* approach, we first implemented a consequence-based algorithm with context. This algorithm is used by the parallel $\mathcal{EL}^{++}$ reasoner ELK [9].
2. In order to compare with the *non-bookkeeping DRed* approach [8], we also implemented an extension of the above approach with the forward-chaining over-deletion used in the non-bookkeeping DRed approach. Such an approach first obtains a set DEL, consisting of all entailments that can be derived from the deleted original axioms. It then effectively over-deletes and re-derives all non-original GCIs OD with the same context as some axiom in DEL.
3. In order to examine the effect of applying our approach with context-based non-bookkeeping DRed, we implemented *our non-bookkeeping approach* on the above one by replacing the context-based re-derivation with our combined forward and backward chaining re-derivation. As we mentioned earlier, this approach will use DEL instead of OD as the over-deleted entailments.
4. In order to compare with the bookkeeping DRed approach, a variant of the first naive reasoning implementation was augmented with the TMS mechanism proposed by [17]. This implementation performs *TMS-based DRed*.
5. In order to examine the effect of applying our approach with TMS-based bookkeeping DRed, we implemented *our TMS-based approach* on the above one by replacing the forward chaining global re-derivation with our re-derivation.

All our implementation used the same completion rules. Hence, they will have the same completion closure for the same input. In order to support reasoning with our evaluation benchmark, our implementations were extended with ABox completion mechanisms. Implementation-wise, this was achieved by internalising ABox axioms with TBox axioms. Such a treatment does not affect the completeness of results on our evaluation benchmark. In order to support the DL used by our evaluation benchmark, our completion rule set extends the **R** rules with the following additional rule to exploit inverse roles in ABox reasoning:

$$\mathbf{R}_{\mathcal{I}} \frac{(a,b) : r, a : C}{b : \exists s.C} : \exists s.C \text{ occurs in } \mathcal{O}, r \sqsubseteq^{*,-} s$$

where $r \sqsubseteq^{*,-} s$ if $r \equiv s^- \in \mathcal{O}$ or $r' \sqsubseteq^* r'$, $r \sqsubseteq^{*,-} s'$ and $s' \sqsubseteq^* s$ and $r \sqsubseteq^* s$ if $r \sqsubseteq s \in \mathcal{O}$, or $r \sqsubseteq^* t$ and $t \sqsubseteq^* s$, or $r \sqsubseteq^{*,-} t$ and $t \sqsubseteq^{*,-} s$. With such extension, the rule set is tractable but it is complete for our evaluation benchmark. Note that in the above formulation, all premise axioms still share a context $\{a\}$. Hence, the extension should not affect the context-based parallelisation of the original rule set.

Test Environment

For preparing the evaluation benchmark, we have used the Lehigh University Benchmark (LUBM) [6] with 10 universities, The University Ontology

Benchmark (UOBM)[1] with 10 universities and Systematised Nomenclature of Medicine - Clinical Terms (SNOMED CT)[2] as experimental datasets. We used el-vira[3] to convert UOBM ontologies to OWL 2 EL ontologies.

All experiments were conducted on 64-bit Ubuntu 14.04 with 3.20 GHz CPU and 10G RAM allocated to JVM. To examine if our approach can reduce the number of over-deleted and/or processed axioms in re-derivation, we were interested in the sizes of the following sets:

Del: deleted original axioms.

DEL: the over-deleted non-original axioms directly or indirectly inferred from Del axioms.

R^*: the completion closure.

DEL_L: the non-original axioms directly or indirectly inferred from Del axioms with the forward chaining over-deletion in the non-bookkeeping DRed approach.

OD_L: the non-original axioms with the same context as some axioms in DEL_L. These axioms, even if preserved, will be re-derived by the forward chaining re-derivation of the non-bookkeeping DRed approach.

$BCRD_L$: the axioms re-derived in the backward chaining re-derivation stage of our non-bookkeeping approach.

OD_T: the over-deleted axioms in the TMS-based DRed approach. These are also the axioms to be processed in the backward chaining re-derivation stage of our TMS-based approach.

$BCRD_T$: the axioms re-derived in the backward chaining re-derivation stage of our TMS-based approach. These are also the axioms to be initialised in L in the forward chaining re-derivation stage of our TMS-based approach. Our implementations are available at https://app.box.com/s/mh81cprp0tgpmjc1qmcjdp00powkcpi9.

We conducted the experiments for $n = 1, 2, 5, 10$, i.e. 2 %, 4 %, 10 % and 20 % of the ABox were updated respectively. For each n, the size of above sets were obtained on the $\lfloor \frac{150}{n} \rfloor$ runs. The reasoning output of the incremental reasoner was the same as the naive reasoner.

We also explored the performance and memory overhead of the TMS. Naive re-computation was performed by the implementation without TMS. In this experiment, we performed tests for 151 times, on the ABoxes $A_1 \cup \cdots \cup A_{50}, A_2 \cup \cdots \cup A_{51}, \ldots, A_{151} \cup \cdots \cup A_{200}$. Each time, we calculated $\%_{initial}$ and $\%_{memory}$. For $T_{deletion}$ and $T_{addition}$, we conducted the experiments for $n = 1, 2, 5, 10$. For each n, the incremental reasoning were performed for $\lfloor \frac{150}{n} \rfloor$ times. The reasoning output of the incremental reasoner was the same as the naive reasoner.

Test Results

The average percentages of $|Del|$, $|R^* \setminus OD_T|$, $|BCRD_T|$, $|DEL_L|$, $|OD_L|$, $|BCRD_L|$ against $|R^*|$ are illustrated in Table 1.

[1] https://www.cs.ox.ac.uk/isg/tools/UOBMGenerator/.
[2] http://www.ihtsdo.org/snomed-ct(2011-Jan.Version).
[3] http://el-vira.googlecode.com.

Table 1. Re-derivation evaluation results (in %)

$\frac{n}{50}$	LUBM				UOBM				SNOMEDCT			
	2	4	10	20	2	4	10	20	2	4	10	20
$\|Del\|/\|R^*\|$	0.55	1.10	2.76	5.52	0.72	1.45	3.62	7.26	0.43	0.86	2.16	4.37
$\|OD_T\|/\|R^*\|$	1.81	3.60	8.88	17.33	1.61	3.22	8.05	16.13	10.66	20.47	45.72	77.58
$\|R^* \setminus OD_T\|/\|R^*\|$	98.2	96.4	91.1	82.7	98.4	96.8	92.0	83.9	89.3	79.5	54.3	22.4
$\|BCRD_T\|/\|R^*\|$	0.52	1.02	2.32	4.02	0.03	0.06	0.14	0.28	0.40	0.73	1.60	2.73
$\|DEL_L\|/\|R^*\|$	3.66	6.35	13.58	23.86	1.83	3.66	9.15	18.33	5.92	11.75	29.10	58.03
$\|OD_L\|/\|R^*\|$	6.47	11.04	22.75	37.88	2.11	4.22	10.57	21.17	5.73	11.42	28.58	58.19
$\|BCRD_L\|/\|R^*\|$	1.70	2.59	4.61	6.72	0.03	0.05	0.13	0.27	0.89	1.66	3.78	6.54

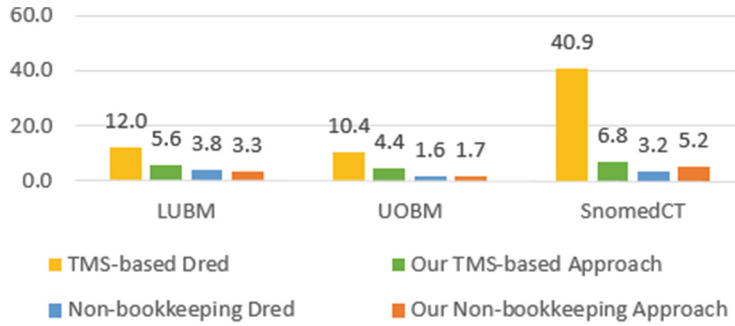

Fig. 4. Time consumption ratio for 2 % Update (in %)

To examine if our approach can be used to achieve efficient incremental reasoning in terms of execution time, in comparison to other approaches, we have conducted experiments using 2 synthetic (LUBM, UOBM) and 1 real-world (SNOMEDCT) datasets to see what would be the ratio of execution time consumed for an update of 2 % in the initial ontology when compared to re-computation. The average values for every approach-dataset pair are illustrated in Fig. 4. We have implemented different algorithms in the environment of TrOWL EL reasoner. Results of experiments are expressed using percentages, instead of absolute values, to proportionally see the effect of different incremental reasoning algorithms and make a comparison between them.

Experiment results regarding the memory overhead are illustrated in Table 2 and Fig. 5.

Observations

1. Because of the nature of *Naive Reasoning*, the cost of time consumed for every small or big update in ontology is always the time of re-computation from scratch(100%). When the update rate is high, this approach can be preferable. But, if the update ratio is as small as 2%, other incremental reasoning techniques become more advantageous. Judging from our experiments, about memory overhead of incremental reasoning, approximately 15% update is the

Table 2. Incremental reasoning evaluation results

$\%_{initial}$	125.89%			
$\%_{memory}$	121.56%			
$\frac{n}{50}$	2%	4%	10%	20%
$\%_{deletion}$	7.37%	14.06%	37.12%	70.21%
$\%_{addition}$	5.94%	15.17%	33.87%	52.44%
$\%_{incremental}$	13.31%	29.23%	70.99%	122.65%

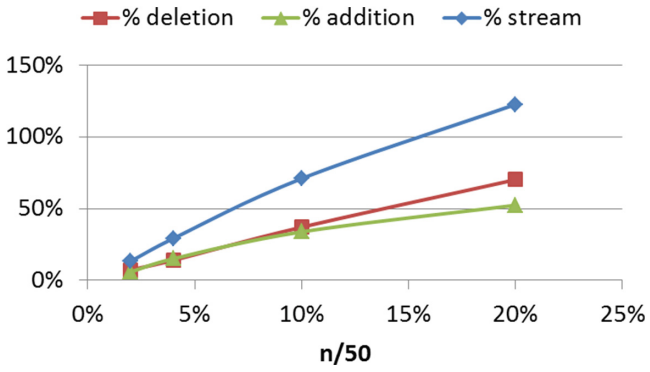

Fig. 5. Incremental reasoning evaluation results

turning point. As illustrated in Table 2 and Fig. 5, up to 15% update in the ontology, incremental reasoning consumes less RAM than naive reasoning, but after that threshold RAM cost of incremental reasoning makes naive reasoning preferable.

2. Using *TMS-based DRed* in a reasoner will impose a performance and memory over-head. The reasoning time was about 25.89 % longer than the same reasoner without TMS. The TMS approach consumed 21.56 % more memory.
3. When *Del* is small, as shown with the ontology SNOMEDCT in Table 1, $BCRD_T$ is much smaller than $R^* \setminus OD_T$ (e.g. 0.40 % v.s. 89.3 % when *Del* is 0.43 % of R^*), indicating that the forward chaining stage in *our TMS-based approach* processes much less axioms than the TMS-based DRed.
   Even when taking into account the cost of the backward chaining, as implied by the size of OD_T, our combined forward and backward chaining approach should still process less axioms than the TMS-based approach.
4. When the size of DEL_L (non-original axioms directly or indirectly inferred from Del axioms) is smaller than the size of OD_L (axioms that are over-deleted and will be re-derived, even if preserved), the *non-bookkeeping DRed* is unnecessarily over-deleting more entailments than necessary. By applying our non-bookkeeping re-derivation approach, the over-deletion in non-bookkeeping approach can be reduced, i.e. over-deleting DEL_L instead of OD_L.

For example, In case of LUBM with 2% update, 3.66% of data, which constitutes the non-original axioms that are inferred from the deleted original axioms, will be selected for over-deletion. Some of this data will be re-derived in forward chain completion. But non-bookkeeping DRed approach chooses a scope of 6.47% of the data for over deletion. By this way 2.81% of data is unnecessarily processed. In this case our non-bookkeeping approach saves the reasoner from ca.77% (2.81/3.66) of unnecessary processing. In case of UOBM with 2% update, the contribution of our non-bookkeeping DRed approach is 15% ((2.11-1.83)/1.83) when compared to non-bookkeeping DRed approach. In case of SNOMEDCT, we don't observe big contribution but nearly same results.

5. *Our non-bookkeeping approach* and *non-bookkeeping DRed* continuously consumed less computation time when compared to other approaches. When interconnections in ontologies increase, performance advantage of them against naive re-computation and global approach becomes more obvious. Increase in the interconnected axioms makes processing of TMS-based Global DRed longer in terms of execution time but does not have that much increase in the processing of them.

To summarise, our combined forward and backward chaining re-derivation technology is very suitable for ontology updating with small scale deletion. It can significantly reduce the re-derivation effort in comparison to the bookkeeping global re-derivation approach. It can reduce the unnecessary over-deletion in comparison to the non-bookkeeping local re-derivation approach. It can also be used to address the completeness issue of the counting approach.

6 Conclusion

In this paper, we presented a novel approach for ontology incremental reasoning. Although we chose the proposed approach is presented in $\mathcal{EL}$, the approach can be used to other completion based algorithms. The motivation of using $\mathcal{EL}$ is due to the effective $\mathcal{EL}$ based approximate reasoning approach [15] implemented in the TrOWL ontology reasoner. Thus we can combine our approach with the approximate reasoning approach for OWL 2 DL incremental reasoning.

Based on a DRed framework, our approach first uses backward chaining to re-derive the over-deleted axioms that can be directly inferred from preserved axioms, and then uses these directly re-derived axioms to initiate forward chaining and re-derive the completion closure of the preserved axioms. This approach can be combined with different over-deletion techniques. It can also be used with or without bookkeeping. The implementation of our approach does not affect the parallelisation or tractability of reasoning and its mechanism is applicable to many consequence-based algorithm. Evaluation results showed that our approach can indeed reduce unnecessary over-deletion and/or re-derivation in a DRed incremental reasoner and can perform efficient incremental reasoning, particularly when the ontology update is of small size in comparison to the ontology, which is where incremental reasoning is mostly needed.

The backward chaining stage of our approach derives the immediate results of the preserved closure. Such an idea has also been exploited in [11] (in their Algorithm 1.3) and [8] (in their Algorithm 4). The difference is that existing approaches derive such immediate results by forward chaining with all the preserved entailments or un-deleted original axioms, which will essentially re-process the entire new closure or the entire broken contexts. Our approach uses backward chaining to avoid the unnecessary processing. Backward chaining can be implemented easily with rule systems. Hence, the original DRed strategy [7], its declarative variant [19] and the ontological adoption of the latter [21] can also exploit such a backward chaining mechanism. Nevertheless, we notice that backward chaining only needs to be performed to re-derive immediate consequence of the preserved partial closure. Hence, expensive recursive full backward chaining can be avoided. Also, our approach only considers a given completion rule set and does not need to generate additional rules from the axioms.

In the future we would like to combine the strengths of different approaches to develop an adaptive incremental reasoning framework, e.g., using TMS to deal with deletion of side condition axioms and contexts to deal with deletion of non-side condition axioms.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
2. Baader, F., Lutz, C., Suntisrivaraporn, B.: Is tractable reasoning in extensions of the description logic EL useful in practice? In: Proceedings of the 2005 International Workshop on Methods for Modalities (M4M–2005) (2005)
3. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: C-SPARQL: SPARQL for continuous querying. In: WWW 2009 (2009)
4. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 1–15. Springer, Heidelberg (2010)
5. Bolles, A., Grawunder, M., Jacobi, J.: Streaming SPARQL - extending SPARQL to process data streams. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 448–462. Springer, Heidelberg (2008)
6. Guo, Y., Pan, Z., Heflin, J.: LUBM: a benchmark for OWL knowledge base systems. Web Semant. Sci. Serv. Agents World Wide Web **3**(2–3), 158–182 (2005)
7. Gupta, A., Mumick, I.S., Subrahmanian, V.S.: Maintaining views incrementally. In: SIGMOD 1993 (1993)
8. Kazakov, Y., Klinov, P.: Incremental reasoning in OWL EL without bookkeeping. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 232–247. Springer, Heidelberg (2013)
9. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK. J. Autom. Reasoning **53**, 1–61 (2013)

10. Klarman, S., Meyer, T.: Prediction and explanation over DL-*Lite* data streams. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) LPAR-19 2013. LNCS, vol. 8312, pp. 536–551. Springer, Heidelberg (2013)
11. Kotowski, J., Bry, F., Brodt, S.: Reasoning as axioms change. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 139–154. Springer, Heidelberg (2011)
12. Lecue, F., Pan, J.Z.: Predicting knowledge in an ontology stream. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013) (2013)
13. Luther, M., Bohm, S., Mobility, S.-A.: An application for stream reasoning. In: Proceedings of 1st International Workshop on Stream Reasoning (SR2009) (2009)
14. Motik, B., Nenov, Y., Piro, R., Horrocks, I.: Incremental update of datalog materialisation: the backward/forward algorithm. In: Proceedings of the 29th National Conference on Artificial Intelligence (AAAI 2015), pp. 1560–1568 (2015)
15. Pan, J.Z., Ren, Y., Zhao, Y.: Tractable approximate deduction for OWL. Artif. Intell. **235**, 95–155 (2016)
16. Parsia, B., Halaschek-Wiener, C., Sirin, E.: Towards incremental reasoning through updates. In: OWL DL, Proceedings of WWW-2006 (2006)
17. Ren, Y., Pan. J.Z.: Optimising ontology stream reasoning with truth maintenance system. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 831–836. ACM (2011)
18. Scharrenbach, T., Urbani, J., Margara, A., Della Valle, E., Bernstein, A.: Seven commandments for benchmarking semantic flow processing systems. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 305–319. Springer, Heidelberg (2013)
19. Staudt, M., Jarke, M.: Incremental maintenance of externally materialized views. In: Vijayaraman, T.M., Buchmann, A.P., Mohan, C., Sarda, N.L. (eds.) Proceedings of the 22th International Conference on Very Large Data Bases (VLDB 1996), 3–6 September 1996, Mumbai, India, pp. 75–86. Morgan Kaufmann (1996)
20. Urbani, J., Margara, A., Jacobs, C., van Harmelen, F., Bal, H.: DynamiTE: parallel materialization of dynamic RDF data. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 657–672. Springer, Heidelberg (2013)
21. Volz, R., Staab, S., Motik, B.: Incrementally maintaining materializations of ontologies stored in logic databases. In: Spaccapietra, S., Bertino, E., Jajodia, S., King, R., McLeod, D., Orlowska, M.E., Strous, L. (eds.) Journal on Data Semantics II. LNCS, vol. 3360, pp. 1–34. Springer, Heidelberg (2005)