Optimising Parallel ABox Reasoning of $\mathcal{EL}$ Ontologies*

Yuan Ren[1], Jeff Z. Pan[1] and Kevin Lee[2]

[1]University of Aberdeen, Aberdeen, UK
[2]NICTA, Australia

Abstract. The success of modern multi-core processors makes it possible to develop parallel ABox reasoning algorithms to facilitate efficient reasoning on large scale ontological data sets. In this paper, we extend a parallel TBox reasoning algorithm for $\mathcal{ELH}_{\mathcal{R}+}$ to a parallel ABox reasoning algorithm for $\mathcal{ELH}_{\perp,\mathcal{R}+}$, which also supports the bottom concept so as to model disjointness and inconsistency. In design of algorithms, we exploit the characteristic of ABox reasoning in $\mathcal{ELH}_{\perp,\mathcal{R}+}$ to improve parallelisation and reduce unnecessary resource cost. Particularly, we separate the TBox reasoning, ABox reasoning on types and ABox reasoning on relations. Our evaluation shows that a naive implementation of our approach can compute all ABox entailments of a Not-Galen$^-$ ontology with about 1 million individuals and 9 million axioms in about 3 minutes.

1 Introduction

Optimisation of reasoning algorithms is one of the core research topics in description logic (DL) study. In the last decades, highly-efficient DL reasoning systems have been implemented with different optimisation technologies. So far, these systems are designed for a single computation core. Reasoning is performed sequentially and can not be parallelised. With the development of modern computing hardware, it is possible and also desired to parallelise reasoning procedures to improve efficiency and scalability.

One direction of parallel reasoning is to use a cluster of multiple computer nodes (or simply, *peers*). In Marvin [9], peers use a divide-conquer-swap strategy for RDFS inference. Weaver and Handler propose a parallel RDFS inference engine [18], in which peers use an ABox partitioning approach. In SAOR [2], peers use optimised template rules for join-free inference in pD* [3]. In DRAGO [13], peers perform OWL DL reasoning under the setting of Distributed Description Logics. A distributed resolution algorithm for $\mathcal{ALC}$ was proposed by Schlicht and Stuckenschmidt [11] and further improved and extended to $\mathcal{ALCHIQ}$ [12]. MapReduce has also been adopted to support ABox reasoning in RDFS [17], pD* [16] as well as justifications in pD* [19], and TBox reasoning in $\mathcal{EL}^+$ [7] (there is no implementation for the $\mathcal{EL}^+$ case yet).

Another direction of parallel reasoning is to use multiple computation cores (or simply, *workers*) in a single computer. Soma and Prasanna [14] propose to use data-partitioning and rule-partitioning in their parallel algorithms for pD*. Liebig and M$\ddot{u}$ller exploit the non-determinism introduced by disjunctions or number restrictions in the $\mathcal{SHN}$ tableau algorithm [6], so that multiple workers can apply expansion rules on

* This paper is an extended version of "Parallel ABox Reasoning of $\mathcal{EL}$ Ontologies" [10].

independent alternatives. Similarly, Meissner [8] proposes parallel expansions of independent branchings in an $\mathcal{ALC}$ tableau and experimented with 3 different strategies. Aslani and Haarslev [1] propose a parallel algorithm for OWL DL classification. Recently, Kazakov et al. [4] present a lock-free parallel completion-based TBox classification algorithm for $\mathcal{ELH}_{\mathcal{R}+}$. They later extend this work to support nominals [5] but the impact on parallelisation has not been reported.

In this paper, we extend the parallel TBox reasoning algorithm [4] for $\mathcal{ELH}_{\mathcal{R}+}$ to a parallel and lock-free ABox reasoning algorithm for $\mathcal{ELH}_{\bot,\mathcal{R}+}$, which also supports the bottom concept so as to model disjointness and inconsistency. We will optimise the parallelisation by separating TBox classification, the computation of types and relations for individuals. We show that our completion rules and algorithms are complete and sound. Our evaluation shows that a naive implementation of our approach can achieve high performance and scalability. Comparing to the original version [10], this paper extends with new optimisations (particularly, Sect. 4.2 and Sect. 4.3) and evaluation regarding ABox reasoning.

The remainder of the paper is organised as follows: In Sect. 2 we introduce background knowledge of DLs $\mathcal{ELH}_{\mathcal{R}+}$ and $\mathcal{ELH}_{\bot,\mathcal{R}+}$, and the parallel $\mathcal{ELH}_{\mathcal{R}+}$ TBox classification algorithm [4]. In Sect. 3 we explain the technical challenges , before presenting the completion rules and parallel ABox reasoning algorithms for $\mathcal{ELH}_{\bot,\mathcal{R}+}$ in Sect. 4. We evaluate our approach in Sect. 5, before we conclude the paper in Sect. 6.

The proof of all lemmas and theorems are included in our online tech report at http://www.box.com/s/3636a703614b65f6cdba.

2 Preliminary

2.1 DL $\mathcal{ELH}_{\mathcal{R}+}$ and $\mathcal{ELH}_{\bot,\mathcal{R}+}$

A *signature* of an ontology $\mathcal{O}$ is a triple $\Sigma_{\mathcal{O}} = (\mathcal{CN}_{\mathcal{O}}, \mathcal{RN}_{\mathcal{O}}, \mathcal{IN}_{\mathcal{O}})$ consisting of three mutually disjoint finite sets of atomic concepts $\mathcal{CN}_{\mathcal{O}}$, atomic roles $\mathcal{RN}_{\mathcal{O}}$ and individuals $\mathcal{IN}_{\mathcal{O}}$. Given a signature, complex concepts in $\mathcal{ELH}_{\bot,\mathcal{R}+}$ can be defined inductively using the $\mathcal{ELH}_{\bot,\mathcal{R}+}$ constructors as in Table 1. $\mathcal{ELH}_{\mathcal{R}+}$ supports all $\mathcal{ELH}_{\bot,\mathcal{R}+}$ constructors except $\bot$. Two concepts C and D are equivalent if they mutually include each other, denoted by $C \equiv D$. An ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, which are finite sets of TBox axioms and ABox axioms, respectively. $\mathcal{ELH}_{\bot,\mathcal{R}+}$ allows all axioms listed in Table 1. $\mathcal{ELH}_{\mathcal{R}+}$ allows all except individual inequalities. The semantics of constructors and axioms are also listed in Table 1. Given an ontology $\mathcal{O}$, we use $\sqsubseteq^*_{\mathcal{O}}$ to represent the reflexive transitive closure of RIs. It is easy to see that in an $\mathcal{ELH}_{\mathcal{R}+}$/ $\mathcal{ELH}_{\bot,\mathcal{R}+}$ ontology $\mathcal{O}$, all of such $\sqsubseteq^*_{\mathcal{O}}$ relations can be computed in polynomial time w.r.t. the size of $\mathcal{O}$. In ABox reasoning, we are particularly interested in ABox materialisation, i.e. finding all $A(a)$ s.t. $a \in \mathcal{IN}_{\mathcal{O}}$, $A \in \mathcal{CN}_{\mathcal{O}}$, $\mathcal{O} \models A(a)$ and all $r(a, b)$ s.t. $a, b \in \mathcal{IN}_{\mathcal{O}}$, $r \in \mathcal{RN}_{\mathcal{O}}$ and $\mathcal{O} \models r(a, b)$. Such results can be very useful for efficient on-line instance retrieval and/or query answering.

2.2 Parallel TBox Classification of $\mathcal{ELH}_{\mathcal{R}+}$ Ontologies

Given an ontology $\mathcal{O}$, TBox classification is a reasoning task that computes all inclusions over atomic concepts in $\mathcal{O}$. Kazakov et. al [4] proposed an approach to parallel

Table 1. $\mathcal{ELH}_{\bot,\mathcal{R}+}$ syntax and semantics

Concepts:		
atomic concept	A	$A^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \mid \exists y.\langle x, y\rangle \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
Roles:		
atomic role	r	$r^{\mathcal{I}}$
TBox Axioms:		
general concept inclusion (GCI):	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion (RI):	$r \sqsubseteq s$	$r^{\mathcal{I}} \sqsubseteq s^{\mathcal{I}}$
role transitivity:	$Trans(t)$	$t^{\mathcal{I}} \times t^{\mathcal{I}} \subseteq t^{\mathcal{I}}$
ABox Axioms:		
class assertion:	$A(a)$	$a^{\mathcal{I}} \in A^{\mathcal{I}}$
role assertion:	$r(a, b)$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in r^{\mathcal{I}}$
individual equality:	$a \doteq b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
individual inequality:	$a \dot{\neq} b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

TBox classification for $\mathcal{ELH}_{\mathcal{R}+}$. They devise a set of completion rules as follows.

$$\mathbf{R}_{\sqsubseteq}\frac{C \sqsubseteq D}{C \sqsubseteq E} : D \sqsubseteq E \in \mathcal{O} \qquad \mathbf{R}^{-}_{\sqcap}\frac{C \sqsubseteq D_1 \sqcap D_2}{C \sqsubseteq D_1; C \sqsubseteq D_2}$$

$$\mathbf{R}^{+}_{\top}\frac{C \sqsubseteq C}{C \sqsubseteq \top} : \top \text{ occurs in } \mathcal{O} \qquad \mathbf{R}^{-}_{\exists}\frac{C \sqsubseteq \exists R.D}{D \sqsubseteq D}$$

$$\mathbf{R}^{+}_{\sqcap}\frac{C \sqsubseteq D_1, C \sqsubseteq D_2}{C \sqsubseteq D_1 \sqcap D_2} : D_1 \sqcap D_2 \text{ occurs in } \mathcal{O}$$

$$\mathbf{R}^{+}_{\exists}\frac{C \sqsubseteq D}{\exists s.C \rightarrow \exists s.D} : \exists s.D \text{ occurs in } \mathcal{O}$$

$$\mathbf{R}_{\mathcal{H}}\frac{D \sqsubseteq \exists r.C,\ \exists s.C \rightarrow E}{D \sqsubseteq E} : r \sqsubseteq^{*}_{\mathcal{O}} s$$

$$\mathbf{R}_{T}\frac{D \sqsubseteq \exists r.C,\ \exists s.C \rightarrow E}{\exists t.D \rightarrow E} : r \sqsubseteq^{*}_{\mathcal{O}} t \sqsubseteq^{*}_{\mathcal{O}} s,\ Trans(t) \in \mathcal{O}$$

In the above rules, $D \rightarrow E$ denotes the special form of GCIs where D and E are both existential restrictions. Given an $\mathcal{ELH}_{\mathcal{R}+}$ ontology $\mathcal{O}$ that has no ABox, these rules infer $C \sqsubseteq D$ iff $\mathcal{O} \models C \sqsubseteq D$ for all C and D such that $C \sqsubseteq C \in \mathbf{S}$ and D occurs in $\mathcal{O}$ [4, Theorem 1], where $\mathbf{S}$ is the set of axioms closed under the above rules. The completion rules are designed in a way that *all premises of each rule have a common concept* (the concept C in each rule), which is called a *context* of the corresponding premise axiom(s). Each context maintains a queue of axioms called *scheduled*, on which some completion rule can be applied, and a set of axioms called *processed*, on which some completion rule has already been applied. *An axiom can only be included in the scheduled queues and/or processed sets of its own contexts.* To ensure that multiple

workers can share the queues and sets without locking them, they further devised a concurrency mechanism, in which: (i) each worker will process a single context at a time and vice versa; (ii) the processing of all axioms in the scheduled queue of a context requires no axioms from the processed sets of other contexts. To realise all these, all contexts with non-empty schedules are arranged in a queue called *activeContexts*. A context can be added into the *activeContexts* queue only if it is not already in the queue.

Here are the key steps of the parallel TBox algorithm:

1. Tautology axiom $A \sqsubseteq A$ for each $A \in \mathcal{CN}_{\mathcal{O}}$ is added to the scheduled queue of A. All active contexts are added into the queue of *activeContexts*.
2. Every idle worker always looks for the next context in the *activeContexts* queue and processes axioms in its scheduled queue.
   (a) Pop an axiom from the scheduled queue, add it into the processed set of the context.
   (b) Apply completion rules to derive conclusions.
   (c) Add each derived conclusion into the scheduled queue of its corresponding context(s), which will be activated if possible.

Before we extend the parallel TBox reasoning algorithm to support ABox reasoning in Sect. 4, we first discuss the challenges to deal with in parallel ABox reasoning.

3 Technical Challenges: Parallel ABox Reasoning

A naive approach to ABox materialisation is to internalise the entire ABox into TBox (i.e., converting assertions of form $C(a)$ into $\{a\} \sqsubseteq C$, and $R(a, b)$ into $\{a\} \sqsubseteq \exists R.\{b\}$) and treat the internalised "nominals" as atomic concepts with TBox classification rules. This is inefficient due to unnecessary computation and maintenance costs. For example, axiom $\{a\} \sqsubseteq \exists r.C$ has C as a context. Thus once derived, it will be maintained in the *processed* set of C, as a possible left premise of Rule $\mathbf{R}_{\mathcal{H}}$ and/or $\mathbf{R}_T$. However, $\mathcal{ELH}_{\bot,\mathcal{R}+}$ does not support nominals, meaning that any corresponding right premise $\exists s.C \rightarrow E$ can always be computed independently from (or before) the derivation of $\{a\} \sqsubseteq \exists r.C$. Therefore it is unnecessary to maintain $\{a\} \sqsubseteq \exists r.C$ in context C because with all possible right premises pre-computed, it can be directly used to trigger all rules.

Even with TBox and ABox reasoning separated, performance and scalability can still be improved: (1) The computation of relations in $\mathcal{ELH}_{\bot,\mathcal{R}+}$ is independent from the computation of types (Lemma 2). Thus when crafting type reasoning rules, relations can be used as side conditions instead of premises. (2) Among all the relations that can be entailed by an $\mathcal{ELH}_{\bot,\mathcal{R}+}$ ontology, some can be trivially computed and do not contribute to type computation. These relations can be easily recovered in retrieval (Lemma 3). (3) Without individual equality, the computation of relations in $\mathcal{ELH}_{\bot,\mathcal{R}+}$ can be perfectly parallelised on an individual basis. This indicates that when computing relations, the concurrency mechanism can be simplified. As we will show, we no longer need to maintain the *scheduled* queue and this improves scalability and performance.

The above optimisations are related to the design of completion rules. For further optimisation on the execution of algorithms, we refer readers to our original paper [10].

4 Approach

4.1 TBox Completion Rules

We first extend the $\mathcal{ELH}_{\mathcal{R}+}$ TBox completion rules to support the bottom concept:

$$\mathbf{R}_{\bot}\frac{D \sqsubseteq \exists r.C, C \sqsubseteq \bot}{D \sqsubseteq \bot}$$

In what follows, we call the set containing the above rule and the $\mathcal{ELH}_{\mathcal{R}+}$ rules in Sect. 2.2 the **R** rule set, which is sound and complete for $\mathcal{ELH}_{\bot,\mathcal{R}+}$ classification:

Lemma 1. *For an $\mathcal{ELH}_{\mathcal{R}+}$ TBox $\mathcal{O}$, let S be any set of TBox axioms closed under the **R** rule set, using $\mathcal{O}$ axioms as side conditions, and $\bot \sqsubseteq \bot \in S$ if $\bot$ occurs in $\mathcal{O}$. Then for any C and D such that $C \sqsubseteq C \in S$, D occurs in $\mathcal{O}$, we have $\mathcal{O} \models C \sqsubseteq D$ iff $C \sqsubseteq D \in S$ or $C \sqsubseteq \bot \in S$.*

The $\leftarrow$ direction is trivial. The $\rightarrow$ direction can be proved with contrapositive. Assuming there are $X \sqsubseteq X \in S$ and Y occurs in $\mathcal{O}$ s.t. $X \sqsubseteq Y \notin S$ and $X \sqsubseteq \bot \notin S$, we construct a model of $\mathcal{O}$ based on S and shows that this model does not satisfy $X \sqsubseteq Y$.

With the **R** rules we can perform TBox reasoning:

Definition 1. ***(TBox Completion Closure)*** *Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}_{\bot,\mathcal{R}+}$ ontology, its TBox completion closure, denoted by $S_{\mathcal{T}}$, is the smallest set of axioms closed under rule set **R**, using $\mathcal{O}$ axioms as side conditions, such that: for all $A \in \mathcal{CN}_{\mathcal{O}}$, $A \sqsubseteq A \in S_{\mathcal{T}}$; $\bot \sqsubseteq \bot \in S_{\mathcal{T}}$ if $\bot$ occurs in $\mathcal{O}$.*

According to Lemma 1, we have $A \sqsubseteq C \in S_{\mathcal{T}}$ or $A \sqsubseteq \bot \in S_{\mathcal{T}}$ for any A and C where A is an atomic concept and C occurs in $\mathcal{T}$. This realises TBox classification.

4.2 Relation Completion Rules

As we pointed out in Sect. 3, relations can be computed independently from types. This is characterised by the following lemma.

Lemma 2. *Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}_{\bot,\mathcal{R}+}$ ontology and $\mathcal{A}_t \subseteq \mathcal{A}$ be the set of all class assertions. Then $\mathcal{O}$ is inconsistent, or for all $r \in \mathcal{RN}_{\mathcal{O}}$, $a, b \in \mathcal{IN}_R$, we have $\mathcal{O} \models r(a, b)$ iff $\mathcal{O} \setminus \mathcal{A}_t \models r(a, b)$.*

Now we present the ABox completion rules for $\mathcal{ELH}_{\bot,\mathcal{R}+}$. Although $\mathcal{ELH}_{\bot,\mathcal{R}+}$ does not support nominals ($\{a\}$), we still denote individuals with nominals since this helps simplify the presentation: (i) ABox rules are more readable, as they have similar syntactic forms to the TBox ones, and (ii) some of the ABox rules can be unified. More precisely, we establish the following mappings as syntactic sugar:

$$C(a) \Leftrightarrow \{a\} \sqsubseteq C, \qquad a \doteq b \Leftrightarrow \{a\} \equiv \{b\},$$
$$a \dot{\neq} b \Leftrightarrow \{a\} \sqcap \{b\} \sqsubseteq \bot, \quad r(a, b) \Leftrightarrow \{a\} \sqsubseteq \exists r.\{b\},$$

Obviously, these mappings are semantically equivalent. In the rest of the paper, without further explanation, we treat the LHS and RHS of each of the above mappings as a *syntactic* variation of each other.

We first present the relation completion rules as follows:

$$\mathbf{AR}^R_{\mathcal{H}} \frac{\{b\} \sqsubseteq \exists r.\{a\}}{\{d\} \sqsubseteq \exists r.\{c\}} : \{a\} \sqsubseteq \{c\}, \{d\} \sqsubseteq \{b\} \in \mathcal{A}$$

$$\mathbf{AR}^R_{\mathcal{T}} \frac{\{b\} \sqsubseteq \exists r.\{a\}}{\{c\} \sqsubseteq \exists s.\{a\}} : \{c\} \sqsubseteq \exists t.\{b\} \in \mathcal{A}, r, t \sqsubseteq^*_{\mathcal{O}} s, Trans(s) \in \mathcal{O}$$

It is worth noting that (1) without individual equality, the $\mathbf{AR}^R_{\mathcal{H}}$ rule is not needed, rendering the $\mathbf{AR}^R_{\mathcal{T}}$ rule perfectly parallelisable. This is an important property because in $\mathcal{ELH}_{\bot,\mathcal{R}+}$ ontologies, individual equality can be easily eliminated by pre-computing equal individuals and using one of them as a representative; (2) relation computation is also independent from TBox completion. We call the reasoning results with the above rules the relation completion closure:

Definition 2. ***(Relation Completion Closure)*** *Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}_{\bot,\mathcal{R}+}$ ontology, its relation completion closure, denoted by $S_{\mathcal{R}}$, is the smallest set of axioms closed under the rules $\mathbf{AR}^R_{\mathcal{H}}$ and $\mathbf{AR}^R_{\mathcal{T}}$, using $\mathcal{O}$ axioms as side conditions, such that $S_{\mathcal{T}} \subseteq S_{\mathcal{R}}$ and $\mathcal{A} \setminus \mathcal{A}_t \subseteq S_{\mathcal{R}}$ (axioms mapped as elaborated before), where $\mathcal{A}_t \subseteq \mathcal{A}$ is the set of all class assertions.*

The soundness and tractability of relation completion closure is quite obvious. Its completeness can be characterised by the following lemma.

Lemma 3. *For any $\mathcal{ELH}_{\bot,\mathcal{R}+}$ ontology $\mathcal{O}$, let $S_{\mathcal{R}}$ be its relation completion closure, then $\mathcal{O}$ is inconsistent, or $\mathcal{O} \models r(a,b)$ only if $\{a\} \sqsubseteq \exists s.\{b\}, s \sqsubseteq^*_{\mathcal{O}} r \in S_{\mathcal{R}}$ for $r \in \mathcal{RN}_{\mathcal{O}}$.*

4.3 Type Completion Rules

Now we present the other ABox completion rules as follows, which should be applied *after* a complete closure $\mathbf{S}_{\mathcal{R}}$ is constructed. In contrast to the **R** rules, the following rules contain concepts $D_{(i)}$ and E that can take multiple forms including nominals. Thus the mapping between ABox and TBox axioms allows us to describe the rules in a more compact manner. Together with the above two relation completion rules, we call all the ABox completion rules the **AR** rules.

The **AR** rules deserve some explanations: There are clear correspondences between the **R** rules and **AR** rules. For example, $\mathbf{AR}_{\sqsubseteq}$ is an ABox counterpart of $\mathbf{R}_{\sqsubseteq}$ except that the context is explicitly a nominal, and TBox results are used as side conditions. Note that directly applying the **R** rules together with the **AR** rules could introduce unnecessary performance overheads such as axiom scheduling, processing and maintenance as we discussed in Sect. 3. In our approach, we separate TBox reasoning, ABox relation computation and ABox type computation. This helps reduce memory usage and computation time.

$$\mathbf{AR}_{\sqsubseteq}\frac{\{a\}\sqsubseteq D}{\{a\}\sqsubseteq E} : D\sqsubseteq E\in S_{\mathcal{R}}\cup\mathcal{A}$$

$$\mathbf{AR}^{*}_{\mathcal{H}}\frac{\{a\}\sqsubseteq \exists r.D}{\{a\}\sqsubseteq E} : \exists s.D\to E\in S_{\mathcal{R}}, r\sqsubseteq^{*}_{\mathcal{O}} s$$

$$\mathbf{AR}^{*}_{T}\frac{\{a\}\sqsubseteq \exists r.D}{\exists t.\{a\}\to E} : \exists s.D\to E\in S_{\mathcal{R}},\ r\sqsubseteq^{*}_{\mathcal{O}} t\sqsubseteq^{*}_{\mathcal{O}} s,\ Trans(t)\in\mathcal{O}$$

$$\mathbf{AR}^{-}_{\sqcap}\frac{\{a\}\sqsubseteq D_1\sqcap D_2}{\{a\}\sqsubseteq D_1;\{a\}\sqsubseteq D_2}$$

$$\mathbf{AR}^{+}_{\sqcap}\frac{\{a\}\sqsubseteq D_1,\{a\}\sqsubseteq D_2}{\{a\}\sqsubseteq D_1\sqcap D_2} : D_1\sqcap D_2 \text{ occurs in } \mathcal{O}$$

$$\mathbf{AR}^{+}_{\exists}\frac{\{a\}\sqsubseteq D}{\exists s.\{a\}\to\exists s.D} : r\sqsubseteq^{*}_{\mathcal{O}} s, \exists r.D \text{ occurs in } \mathcal{O}$$

$$\mathbf{AR}_{\bot}\frac{\{a\}\sqsubseteq \bot}{\{b\}\sqsubseteq \bot} : \{b\}\sqsubseteq\exists r.\{a\}\in S_{\mathcal{R}}$$

$$\mathbf{AR}^{*}_{\bot}\frac{\{a\}\sqsubseteq \exists r.D}{\{a\}\sqsubseteq \bot} : D\sqsubseteq\bot\in S_{\mathcal{R}}$$

$$\mathbf{AR}_{\mathcal{H}}\frac{\exists s.\{a\}\to E}{\{b\}\sqsubseteq E} : r\sqsubseteq^{*}_{\mathcal{O}} s, \{b\}\sqsubseteq\exists r.\{a\}\in S_{\mathcal{R}}$$

$$\mathbf{AR}_{T}\frac{\exists s.\{a\}\to E}{\exists t.\{b\}\to E} : r\sqsubseteq^{*}_{\mathcal{O}} t\sqsubseteq^{*}_{\mathcal{O}} s,\ Trans(t)\in\mathcal{O}, \{b\}\sqsubseteq\exists r.\{a\}\in S_{\mathcal{R}},$$

Now we defined the closure of applying all the rules:

Definition 3. ***(Ontology Completion Closure)*** *Let $\mathcal{O}=(\mathcal{T},\mathcal{A})$ be an $\mathcal{ELH}_{\bot,\mathcal{R}+}$ ontology, its ontology completion closure, denoted by* **S***, is the smallest set of axioms closed under the* ***AR*** *rule set, with $\mathcal{O}$ axioms as side conditions, such that$S_{\mathcal{R}}\subseteq\mathbf{S}$; $\mathcal{A}\subseteq\mathbf{S}$ (axioms mapped as elaborated at the beginning of this section); and for all $a\in\mathcal{IN}_{\mathcal{O}}$, $\{a\}\sqsubseteq\{a\}\in X_{\mathcal{O}}$, and $\{a\}\sqsubseteq\top$ if $\top$ occurs in $\mathcal{O}$.*

Together the **AR** rules are also complete, sound and tractable. The soundness and tractability of rules are quite obvious. The completeness on ABox materialisation can be shown by the following Theorem:

Theorem 1. *For any $\mathcal{ELH}_{\bot,\mathcal{R}+}$ ontology $\mathcal{O}=(\mathcal{T},\mathcal{A})$, we have either there is some $\{x\}\sqsubseteq\bot\in\mathbf{S}$, or*

1. *$\mathcal{O}\models D(a)$ only if $\{a\}\sqsubseteq D\in\mathbf{S}$ for D occurs in $\mathcal{O}$;*
2. *$\mathcal{O}\models r(a,b)$ only if $\{a\}\sqsubseteq\exists s.\{b\}, s\sqsubseteq^{*}_{\mathcal{O}} r\in\mathbf{S}$ for $r\in\mathcal{RN}_{\mathcal{O}}$.*

The result regarding roles is quite obvious due to Lemma 3 and item 1 in Def. 3. The type part can be proved by contrapositive. Assuming there is no $\{x\}\sqsubseteq\bot\in\mathbf{S}$, it's obvious that the TBox $\mathcal{T}$ has a model. We can construct such a model based on $S_{\mathcal{T}}$ and

shows that it can be extended to a model of $\mathcal{O}$ based on $\mathbf{S}$, such that this model entails a said class assertion only if it is in $\mathbf{S}$. Full proof can be found in our technical report.

As we can see, the **AR** rules also preserve the feature that all premises of each rule have a same common part as the context. Therefore, they still enjoy the lock-free feature in reasoning. In later sections, we will further elaborate this point.

4.4 Parallel Algorithms

In this section, we present the parallel algorithm corresponding to the $\mathcal{ELH}_{\bot,\mathcal{R}+}$ completion rules. We reuse some notions such as *context*, *activeContexts* queue, *scheduled* queue and *processed* set from the original TBox algorithm for $\mathcal{ELH}_{\mathcal{R}+}$ [4] to realise the lock-free mechanism. Axioms are indexed w.r.t. corresponding contexts.

The revised saturation algorithm (Algorithm 1) is presented as follows. The saturation of an ontology is realised by first performing saturation of the non-class assertion axioms, the output of which (i.e., $S_{\mathcal{R}}$) is then used in the saturation of the class assertions. This yields $\mathbf{S}$, which satisfies Theorem 1. This result contains all entailed class assertions, and all role assertions can be easily retrieved. All necessary tautology axioms must be added to the input prior to saturation. For TBox, we add axioms of the form $C \sqsubseteq C$ for all concepts C such that $C \in \mathcal{CN}_{\mathcal{O}} \cup \{\bot\}$. Similarly, for type computation we add $\{a\} \sqsubseteq \{a\}$ for all individuals $a \in \mathcal{IN}_{\mathcal{O}}$.

Algorithm 1: saturate(input): saturation of axioms under inference rules

```
   Input: input (the set of input axioms)
   Result: the saturation of input is computed in context.processed
1  activeContexts ← ∅;
2  axiomQueue.addAll(input);
3  loop
4      axiom ← axiomQueue.pop();
5      if axiom = null then break;
6      for context ∈ getContexts(axiom) do
7          context.scheduled.add(axiom);
8          activeContexts.activate(context);
9  loop
10     context ← activeContexts.pop();
11     if context = null then break;
12     loop
13         axiom ← context.scheduled.pop();
14         if axiom = null then break;
15         process (axiom);
16     context.isActive ← false;
17     if context.scheduled ≠ ∅ then activeContexts.activate(context);
```

In the saturation (Algorithm 1), the $activeContexts$ queue is initialised with an empty set (line 1), and then all input axioms are added into an *axiomQueue* (line 2). After that, two main loops (lines 3-8 and lines 9-17) are sequentially parallelised.

In the first main loop, multiple workers independently retrieve axioms from the *axiomQueue* (line 4), then get the contexts of the axioms (line 6), add the axioms into corresponding *scheduled* queues (line 7) and activate the contexts. In the second main loop, multiple workers independently retrieve contexts from the $activeContexts$ queue (line 10) and process its *scheduled* axioms (line 15). Once $context.scheduled$ is empty, $context.isActive$ is set to $false$ (line 16). A re-activation checking is performed (line 17) in case other workers have added new axioms into $context.scheduled$ while the last axiom is being processed (between line 14 and line 16). This procedure will continue until the $activeContexts$ queue is empty.

The $getContext()$ method returns the context of an axiom, depending on the form of the axiom. In the following list, the concept C or $\{a\}$ is the context.

$$C \sqsubseteq D \mid D \sqsubseteq \exists r.C \mid \exists s.C \rightarrow E \mid D \sqsubseteq \exists r.\{a\} \mid \{a\} \sqsubseteq D$$

To activate a context, an atomic boolean value $isActive$ is associated with each context to indicate whether the context is already active. A context is added into the $activeContexts$ queue only if this value is $false$, which will be changed to $true$ at the time of activation. The $process()$ method covers items 2.(a), 2.(b), 2.(c) shown at the end of Sect. 2.2. We match the form of input axiom and check whether it has been processed before; if not it will be added into the $processed$ set of the context. Based on the form of axiom, applicable completion rules can be determined. Meanwhile, checking if the conclusion is already in corresponding context's processed set can be performed. Once a completion rule has been applied, the conclusion axioms and their forms are determined. Once a conclusion is derived, its contexts and whether they are definitely the same as the current context are determined. The conclusion axioms can directly be added into corresponding $scheduled$ queues.

5 Evaluation

We implemented our algorithms in our PEL reasoner (written in JAVA). To evaluate its performance we use the Amazon Elastic Computer Cloud (EC2) High-Memory Quadruple Extra Large Instance. It has 8 virtual cores with 3.25 EC2 units each, where each EC2 unit "provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor". The instance has 60 GB memory allocated to JVM. Our test cases include a real world TBox NotGalen with generated ABox. NotGalen$^-$ is extracted from an earlier version of Galen (http://www.opengalen.org/) by removing functional role assertions. It contains a moderate-size TBox with 2748 classes, 413 roles and no ABox. To populate the ontology we use the SyGENiA system [15] to generate ABoxes for a small part of the Galen ontology and combine the generated ABoxes of different sizes with the NotGalen$^-$ TBox. In this way, we have test ontologies with larger and larger number of individuals, denoted by NGS-1, NGS-5, NGS-10 etc. Such ABoxes are not completely random because, as generated by SyGENiA, they cover axioms that can lead to all possible sources of incompleteness w.r.t. a TBox and certain query (in our evaluation, we query for instances of $PlateletCountProcedure$). Being able to handle such ABoxes means that the reasoner won't miss any result when dealing with any real-world ABoxes.

For each ontology, we perform ABox materialisation. Note that not all the relations have to be computed in reasoning, but only the necessary ones as indicated in Theorem 1. Nevertheless all relations can be retrieved very easily if needed. Results of our implementation PEL are presented in Table 2. The time shown in our evaluation is the overall computation time. Time unit is second.

Table 2. Results of PEL (in sec) for scalability tests

Ontology	$\|\mathcal{IN}\|$	$\|\mathcal{A}\|$	1 worker	2 workers	4 workers	6 workers
NGS-1	4309	8000	1.378	0.931	0.789	0.733
NGS-5	62639	119852	4.97	4.108	3.746	3.709
NGS-10	211244	437542	16.276	14.196	7.867	6.634
NGS-20	596616	1642664	60.314	52.381	53.559	36.599
NGS-30	865790	3541822	127.517	87.141	83.855	81.964
NGS-40	970925	6036497	180.494	148.802	101.614	105.231
NGS-50	995932	9024356	247.547	204.713	192.055	201.127

From the comparison between different numbers of workers in Table 2 we can see that multiple parallel workers can indeed improve the reasoning performance, even when the ontology contains complex TBox and very large number of individuals. In general, the performance improves when more and more workers are used. With more than 4 workers, the performance may decrease on very large ABoxes. We believe one of the potential reasons is that although the CPU cores can work in parallel, the RAM bandwidth is limited and RAM access is still sequential. In relatively "light-weight" ABox reasoning with large ABox, the RAM access will be enormous and very often so that multiple workers will have to compete for RAM access. This makes memory I/O a potential bottleneck of parallelisation and wastes CPU cycles. Another potential reasoner is that with such a large Jave heapsize, the memory management of the JVM will take quite some time. A better management of memory will be an important direction of our future work.

6 Conclusion

In this paper we extended early related work to present a parallel ABox reasoning approach to $\mathcal{ELH}_{\bot,\mathcal{R}+}$ ontologies. We proposed new completion rules to improve efficiency and scalability of parallel ABox reasoning, and showed that they are complete and sound for ABox reasoning. Our evaluation shows that ABox reasoning can benefit from parallelisation, even for very large ABoxes. In future, We would like to continue the work in two directions. One is to develop distributed algorithms so that the memory cost can also be distributed. The other is to develop target-driven materialisation instead of full materialisation to reduce memory cost.

Acknowledgement

Yuan Ren and Jeff Z. Pan are partially funded by the EU FP7 K-Drive project.

References

1. Aslani, M., Haarslev, V.: Parallel TBox classification in description logics – first experimental results. In: Proceeding of ECAI2010 (2010)
2. Hogan, A., Pan, J.Z., Polleres, A., Decker, S.: SAOR: Template rule optimisations for distributed reasoning over 1 billion linked data triples. In: Proceedings of International Semantic Web Conference. pp. 337–353 (2010)
3. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. J. Web Sem. 3(2-3), 79–115 (2005)
4. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of $\mathcal{EL}$ ontologies. In: Proceedings of ISWC'11. LNCS, vol. 7032. Springer (2011)
5. Kazakov, Y., Krötzsch, M., Simančík, F.: Practical reasoning with nominals in the $\mathcal{EL}$ family of description logics. In: Proceedings of KR'12 (2012)
6. Liebig, T., Müller, F.: Parallelizing tableaux-based description logic reasoning. In: Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems - Volume Part II. pp. 1135–1144. Springer-Verlag, Berlin, Heidelberg (2007)
7. Maier, R.M.F., , Hitzler, P.: A MapReduce algorithm for EL+. In: Proc. of International Worshop of Description Logic (DL2010) (2010)
8. Meissner, A.: Experimental analysis of some computation rules in a simple parallel reasoning system for the $\mathcal{ALC}$ description logic. Applied Mathematics and Computer Science 21(1), 83–95 (2011)
9. Oren, E., Kotoulas, S., Anadiotis, G., Siebes, R., ten Teije, A., van Harmelen, F.: Marvin: Distributed reasoning over large-scale semantic web data. Web Semant. 7, 305–316 (2009)
10. Ren, Y., Pan, J.Z., Lee, K.: Parallel ABox reasoning of EL ontologies. In: Proc. of the First Joint International Conference of Semantic Technology (JIST 2011) (2011)
11. Schlicht, A., Stuckenschmidt, H.: Distributed resolution for ALC. In: Description Logics Workshop (2008)
12. Schlicht, A., Stuckenschmidt, H.: Distributed resolution for expressive ontology networks. In: Proceedings of RR'09. pp. 87–101. Springer-Verlag, Berlin, Heidelberg (2009)
13. Serafini, L., Tamilin, A.: Drago: Distributed reasoning architecture for the semantic web. In: ESWC2005. pp. 361–376. Springer (2005)
14. Soma, R., Prasanna, V.K.: Parallel inferencing for owl knowledge bases. In: Proceedings of the 37th International Conference on Parallel Processing. pp. 75–82. ICPP '08, IEEE Computer Society, Washington, DC, USA (2008)
15. Stoilos, G., Cuenca Grau, B., Horrocks, I.: How incomplete is your semantic web reasoner? In: Proc. of AAAI 10. pp. 1431–1436. AAAI Publications (2010)
16. Urbani, J., Kotoulas, S., Maassen, J., Van Harmelen, F., Bal, H.: WebPIE: A web-scale parallel inference engine using MapReduce. Web Semant. 10, 59–75 (Jan 2012)
17. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable distributed reasoning using MapReduce. In: Proceedings of the ISWC '09. LNCS, vol. 5823. Springer (2009)
18. Weaver, J., Hendler, J.A.: Parallel materialization of the finite RDFS closure for hundreds of millions of triples. In: Proc. of ISWC 2009. LNCS, vol. 5823, pp. 682–697. Springer (2009)
19. Wu, G., Qi, G., Du, J.: Finding all justifications of OWL entailments using TMS and MapReduce. In: Proc. of CIKM2011 (2011)